**TP – Sistemas Distribuídos**

**Aplicação de técnicas de Sistemas Distribuídos**

**Usando Threads, Sockets e Datagrams**

Fernando H. Pimenta

Renan Bicalho Silva

Ouro Preto, 16 de novembro de 2014

Fernando H. Pimenta

12.2.4189

Renan Bicalho Silva

10.2.4020

**Sistemas Distribuídos**

**Trabalho Prático III**

Trabalho apresentado ao professor

Dr. Joubert de Castro Lima

Referente à disciplina de

Sistemas Distribuídos (BCC362).

Ouro Preto, 16 de novembro de 2014

Sumário:

1 Introdução 4

1.1 Objetivos 4

1.2 Considerações iniciais 4

1.3 Descrição do trabalho 4

2 Visão Geral 4

2.1 Descrição 4

2.2 Arquitetura 5

3 Classes 6

3.1 Classe main 6

3.2 Classe Thread\_Host 9

3.3 Classe Thread\_Server 9

3.4 Classe Thread\_Accept 10

4 Experimentos 10

5 Resultados 10

6 Conclusão 11

7 Codigo fonte 12

# Introdução

## Objetivos

* Construir uma máquina que ordene números inteiros num intervalo de 10 milhões
* Implementar um sistema de Multicomputer que utilize threads, sockets e datagrams
* Implementar um método de ordenação

## Considerações iniciais

* Ambiente de desenvolvimento do código fonte: Eclipse Mars 4.5.
* Linguagem utilizada: Linguagem Java.
* Ambiente de desenvolvimento da documentação: Microsoft Word

## Descrição do trabalho

Nesse trabalho iremos implementar algoritmo para extrair dados de um arquivo binário e ordená-los de forma distribuídas em várias máquinas multicore usando threads, sockets. Iremos ordenar arquivos de 1GB, 10GB, 50GB e 100GB utilizando várias máquinas do laboratório do Decom, o com 30.

# Visão Geral

Esta seção tem como objetivo fazer uma explicação mais detalhada sobre o problema e a solução propostos para o primeiro trabalho prático de Sistemas Distribuídos.

## Descrição

Apesar dos grandes avanços da tecnologia nos dias atuais ainda temos algumas limitações, uma delas e em relação a trabalhar com dados massivos em várias máquinas. Uma solução prática veio através de *sockets*. *Socket* é a interface entre a camada de aplicação e a de transporte dentro de uma máquina, ou seja, o elo entre os processos do servidor e do cliente. Através desses conceitos, construiremos um sistema Multicomputer e multicore. Assim, utilizaremos essa ideia para ordenar arquivos grandes contendo números gigantescos. A ideia principal é que cada host que esteja conectado na rede seja responsável por uma parcela dos números, mas todos iram extrair números e separar seus intervalos de acordo com o numero de hosts, então, cada host irá enviar pela rede para o host destinatário o intervalo pertencente ao mesmo, logo, ele receberá de outros hosts o intervalo pertencente a ele. Assim, ele irá juntar essas parcelas em um único vetor que então irá ordená-lo, tendo assim uma ordenação global, lembrando que como no trabalho anterior, cada host usará todos seus núcleos. Para isso, usamos threads, e para rede usamos sockets.

## Arquitetura

Nesta seção a arquitetura do sistema será apresentada através de diagrama UML.

### Classe principal

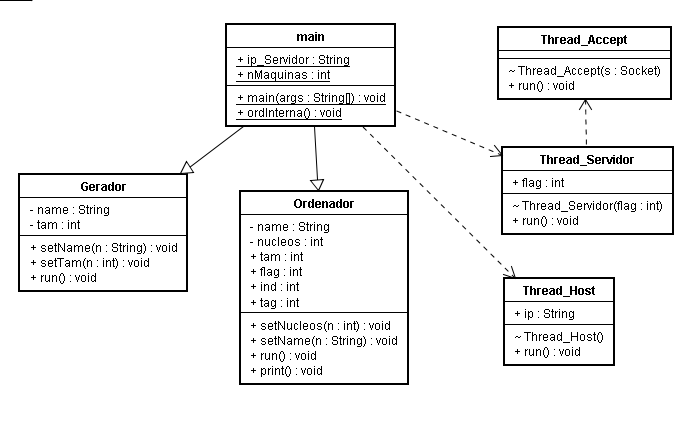


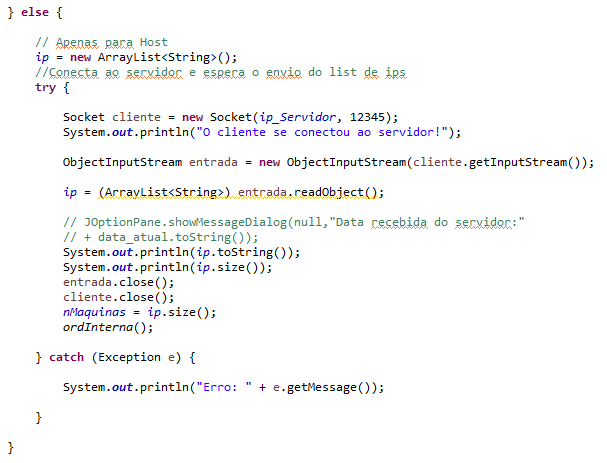
Figura 1: Classes

# Classes

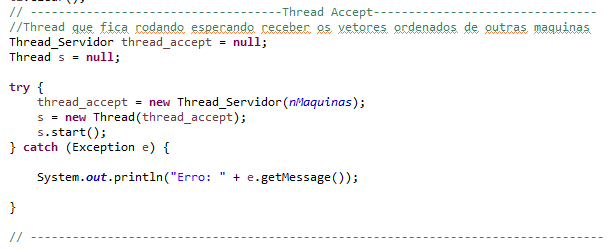
## Classe main

### 

Ao executarmos o programa, ele perguntará se desejamos configurar um servidor. Caso selecionemos “sim” cairemos no “if”. Inicialmente ele aguarda um numero N de computadores, então ele junta o ip de todos os hosts e envia para cada, um List<String> contendo todos os ips dos hosts.

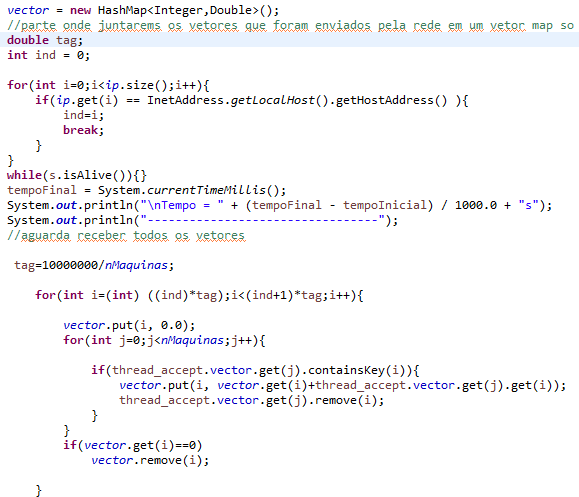


Neste else, caso não configuremos um servidor, conectamos ao ip do mesmo e aguardamos que ele envie um List<String> que conterá os ips dos hosts em rede.

  
Neste trecho de código, subimos uma thread do tipo “Thread\_Servidor” que ficará escutando a porta, esperando os hosts enviarem os vetores de números .

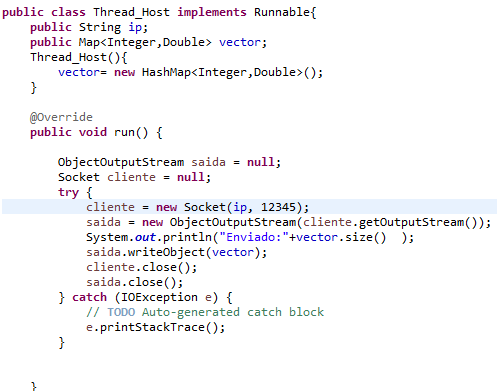


Após separar os intervalos dos numeros (Assunto do trabalho anterior), enviaremos para cada host o seu devido intervalo. Esse envio ocorrerá através de Threads do tipo Thread\_Host.



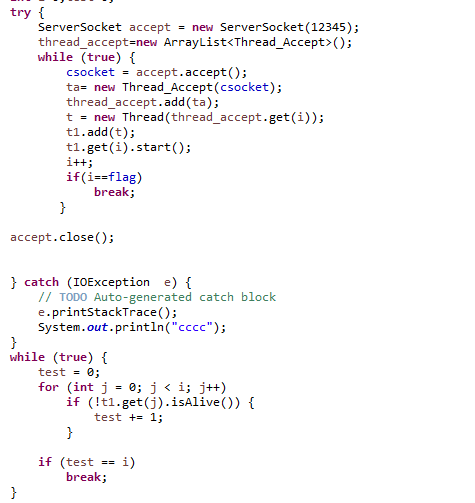
Após enviar e receber os intervalos, cada host contem o seus vetores com seu intervalo, a tarefa agora e juntar esses vetores colocando no Map vector.

## Classe Thread\_Host



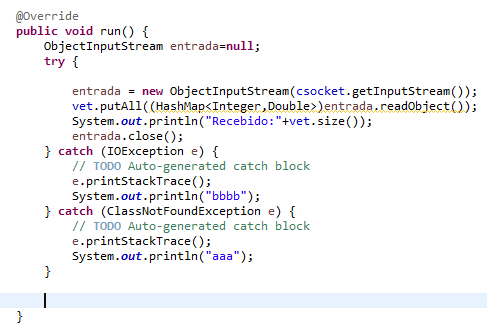
Essa Thread enviará para seu ip destinatário os números no intervalo daquele host.

## Classe Thread\_Server



A Classe thread\_Servidor trata de receber os vetores de números, de modo que, para cada “accept”, ela gera uma nova thread do tipo Thread\_Accept.

## Classe Thread\_Accept



Essa Thread trata de receber os vetores que seram mandados pela rede.

# Experimentos

Para avaliarmos os benefícios que o uso de sockets pode trazer, serão executados uma bateria de experimentos da seguinte forma:

Teste:

No laboratório variamos o tamanho do arquivo de entrada 10GB, 50GB e 100GB.

# Resultados

Os resultados obtidos foram realizados em diferentes máquinas que se localizam no laboratório COM30 pertencente ao DECOM - UFOP. As máquinas utilizadas possuem 8 núcleos de processamento com 16GB de memoria RAM. Na tabela abaixo podemos observar o tempo gasto de ordenação de cada vetor referente ao número de hosts com o envio pela rede.

Tabela 1: Teste para 1GB

|  |  |
| --- | --- |
| Quantidade (Hosts) | Tempo de execução (Segundos) |
| 5 | 15.0 |
| 8 | 6.8 |
| 10 | 1.3 |

Tabela 2: Teste para 10GB

|  |  |
| --- | --- |
| Quantidade (Hosts) | Tempo de execução (Segundos) |
| 5 | 45.6 |
| 8 | 34.26 |
| 10 | 20.41 |

Tabela 3: Teste para 50GB

|  |  |
| --- | --- |
| Quantidade (Hosts) | Tempo de execução (Segundos) |
| 5 | ------ |
| 8 | 234.25 |
| 10 | 212.530 |

Tabela 4: Teste para 100GB

|  |  |
| --- | --- |
| Quantidade (Hosts) | Tempo de execução (Segundos) |
| 5 | ------- |
| 8 | -------- |
| 10 | 713.342 |

# Conclusão

Concluímos que o uso de um sistema distribuído Multicomputer e multicore com o uso de múltiplas máquinas trabalhando em paralelo minimiza o tempo de ordenação, quando trabalhamos com uma quantidade muito grande de dados, e, quanto mais *hosts* e *cores* usamos, mais dividimos as tarefas para cada um tornando, como já dito, o tempo menor. Apesar da eficiência, ainda hoje existe problemas grandes e complicados de várias extensões, mas talvez e provavelmente em um futuro próximo, este paradigma poderá encontrar a solução ótima, ou próximo disso.

Podemos concluir também que este trabalho foi uma ótima forma de colocar em prática tudo que vimos durante as aulas de Sistemas Distribuídos, uma vez que vários conceitos e métodos foram aplicados a fim de construir o sistema proposto. Isto proporciona uma melhor compreensão da ementa da disciplina, através da geração de uma visualização mais prática do funcionamento dos sistemas distribuídos Multicomputer e multicore.

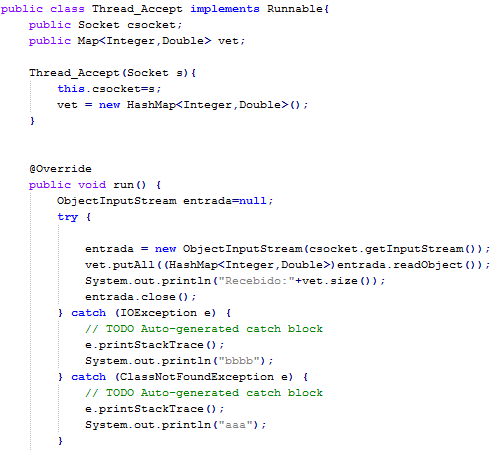
# Código Fonte

## Classe Ordenador

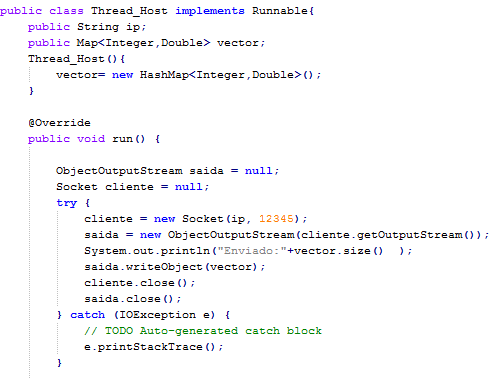


## 

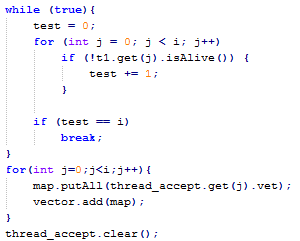
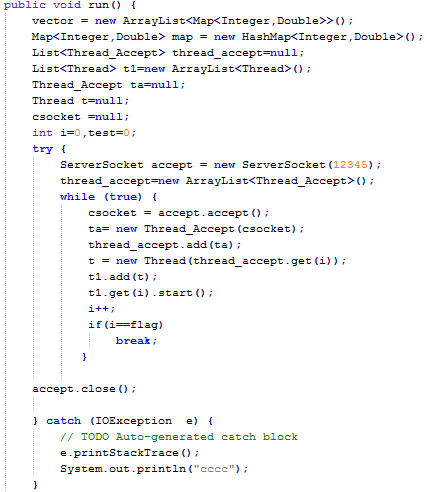
## Classe Thread\_Accept



## Classe Thread\_Host



## Classe Thread\_Servidor



## Main

