GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Aprendizado de Máquina em Plataformas de Processamento Distribuído de Fluxo: Análise e Detecção de Ameaças em Tempo Real

Martin Andreoni Lopez (UFRJ), Igor Jochem Sanz (UFRJ), Antonio G. P. Lobato (UFRJ), Diogo M. F. Mattos (UFF) e Otto Carlos M. B. Duarte (UFRJ)

XXXVI Simpósio Brasileiro de Redes de Computadores

09/05/2018 – Campos do Jordão

Programa de Engenharia Elétrica - PEE/COPPE/UFRJ
Universidade Federal do Rio de Janeiro

COPPE 50 UFRJ

SEGURANÇA CIBERNÉTICA

UM DOS PRINCIPAIS PROBLEMAS DA HUMANIDADE

PARTE I

Introdução e Conceitos Fundamentais de Processamento de Fluxo

Botnet Mirai
Negação de Serviço

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Botnet - exército de zumbis - Ataque distribuído de negação de serviço - *Distributed Denial of Service* (DDoS)
- Outubro 2016 - outubro de 2016, o Mirai botnet atacou o serviço DynDNS, que derrubou Reddit, Twitter e outros grandes sítios
- Mais de 300 mil dispositivos comprometidos
- Paras Jha e Josiah White criadores do código fonte do Mirai desenvolveram a botnet para fraudar o número de cliques de um sítio na Universidade de Rutgers

http://gizmodo.uol.com.br/botnet-mirai-criadores-culpados/

Botnet Mirai
Negação de Serviço

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Botnet - exécito de zumbis - Ataque distribuído de negação de serviço - *Distributed Denial of Service* (DDoS)
- Outubro 2016 - outubro de 2016, o **Mirai botnet atacou o serviço DynDNS, que derrubou Reddit, Twitter e outros grandes sítios**
- **Mais de 300 mil dispositivos comprometidos**
- Paras Jha e Josiah White criadores do código fonte do Mirai desenvolveram a botnet para fraudar o número de cliques de um sítio na Universidade de Rutgers

http://gizmodo.uol.com.br/botnet-mirai-criadores-culpados/

COPPE UFRJ

Botnet Mirai – Ataque de DDoS

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Em setembro 2016 o maior ataque de DDoS alcançou **mais de 1 Tera bit por segundo**

Octave Klaba é o presidente da empresa francesa de hospedagem digital OVH

COPPE UFRJ

Botnet Mirai – Ataque de DDoS

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Em setembro 2016 o maior ataque de DDoS alcançou **mais de 1 Tera bit por segundo**

Octave Klaba é o presidente da empresa francesa de hospedagem digital OVH

COPPE UFRJ

Botnet Mirai – Ataque de DDoS

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Octave Klaba / Oles
@olesovhcom

Follow

Last days, we got lot of huge DDoS. Here, the list of "bigger that 100Gbps" only. You can see the simultaneous DDoS are close to 1Tbps !

Hacked cameras (CCTV system) used by attackers to carry 1Tbps DDoS attack!

Botnet Mirai – Ataque de DDoS

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Botnet Mirai – Ataque de DDoS

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Octave Klaba / Oles
@olesovhcom

Follow

```
log /home/vac/logs/vac.log-last | egrep "pps\|.....
bps" | awk '{print $1,$2,$3,$6}' | sed "s/ /|/g" | cut -f
1,2,3,7,8,10,11 -d '|' | sed "s/.....bps/Gbps/" | sed
"s/.....pps/Mpps/" | cut -f 2,3,4,5,6,7 -d ":" | sort | g
```

Here, the

Mais de 145.000 câmeras IP geraram mais de 1 Tb/s de ataques distribuídos de negação de serviço (DDoS)

```
Sep|20|12:48:17|tcp_ack|49Mpps|572Gbps
Sep|21|05:09:42|tcp_ack|32Mpps|144Gbps
Sep|21|20:21:37|tcp_ack|22Mpps|122Gbps
Sep|22|00:50:57|tcp_ack|16Mpps|191Gbps
You have new mail in /var/mail/root
```

UFRJ

Ataques DDoS de **1,35 Tb/s**
28 de fevereiro de 2018

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

GitHub knocked briefly offline by biggest DDoS attack ever

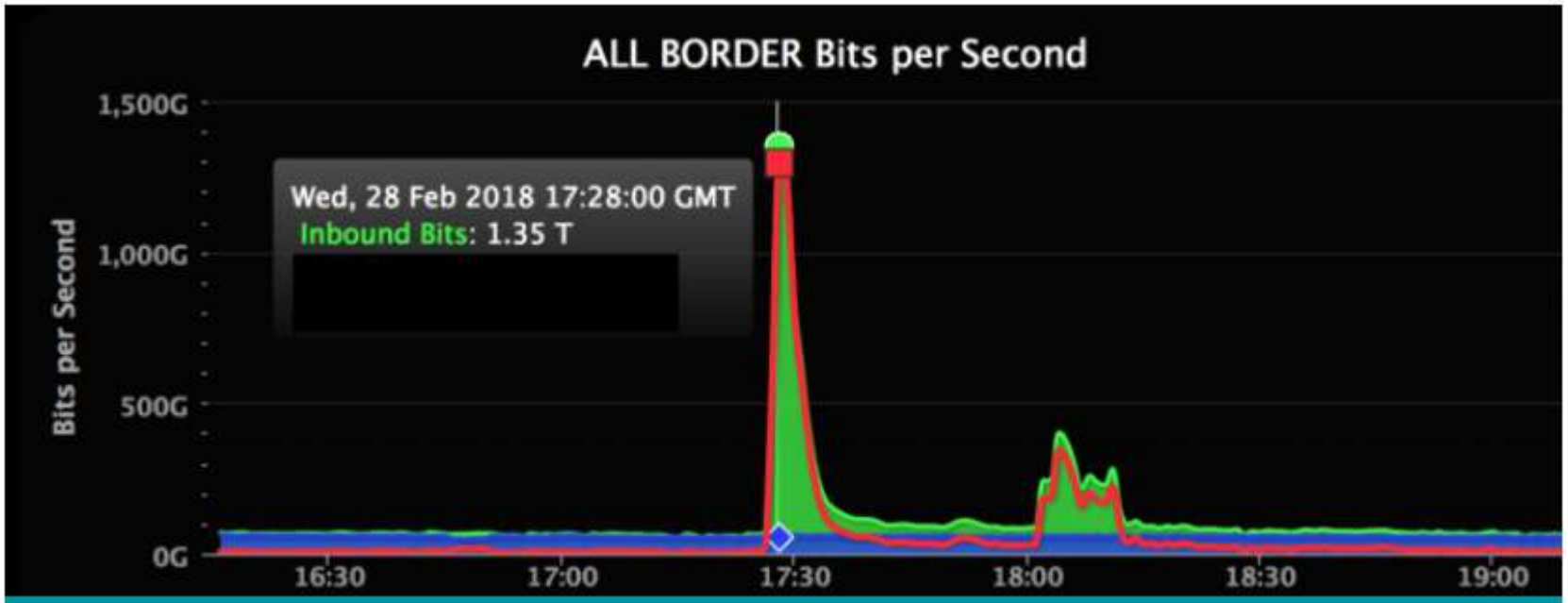
At its peak, inbound traffic reached a staggering 1.35 terabits per second (Tbps), outflanking the previously record-setting assault of 1 Tbps at French web hosting provider OVH in September 2016.

COPPE UFRJ

Ataques DDoS de **1,35 Tb/s**
28 de fevereiro de 2018

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

GitHub knocked briefly offline by biggest DDoS attack ever



Inbound traffic to GitHub during the DDoS attack (source: GitHub, Akamai)

Brasil como alvo de DDoS em 2017

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Quinto país** classificado como alvo em número de ataques de DDoS
- **264.900 ataques no ano** de 2017, 735 por dia e 30 por hora
- **34,9% originados no próprio Brasil,** 30,3% nos EUA, 17,8% no Canadá e 17,8% no Reino Unido

13th annual Worldwide Infrastructure Security Report by NETSCOUT Arbor

https://www.zdnet.com/article/brazil-hit-by-30-ddos-attacks-per-hour-in-2017/

COPPE UFRJ

WannaCry – Vírus de resgate (*Ransomware*)

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

"Ooops, your files have been encrypted!"

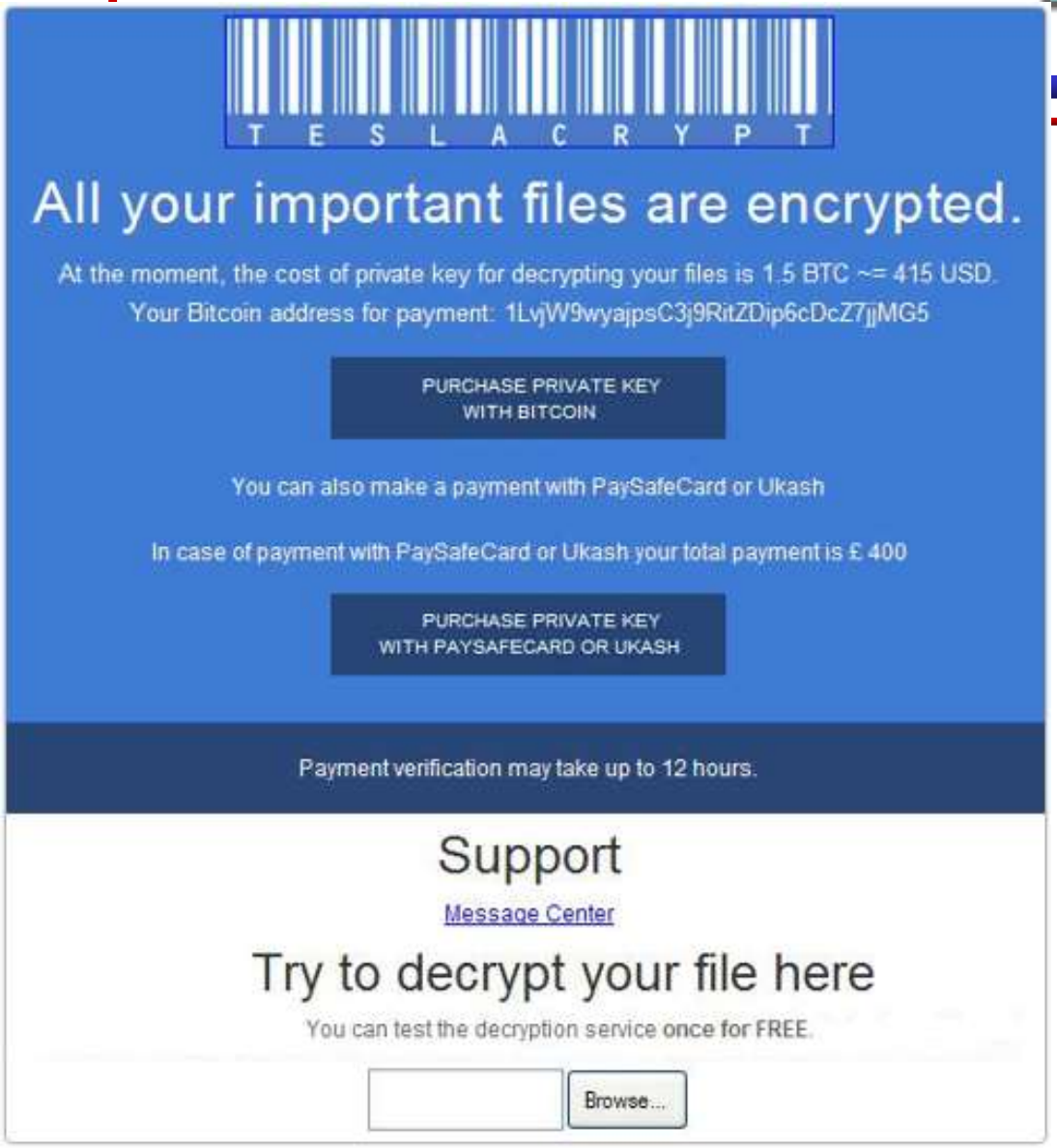
COPPE UFRJ

WannaCry – Vírus de resgate

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Vírus de resgate (*ransomware*) - Ação - encripta o computador e solicita resgate (*ransom*) em bitcoins.
- Cripto-ransomware que afeta o Microsoft Windows, com difusão em larga escala iniciada em 12 de maio de 2017 através *physhing* **infectando mais de 230.000 sistemas**
- Organizações como a Telefonica e o Serviço Nacional de Saúde britânico foram afetadas, juntamente com outras operadoras de telecomunicações, empresas de transportes, organizações governamentais, bancos e universidades.

Teslacript

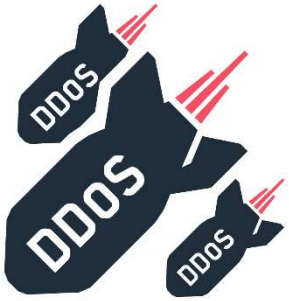

T E S L A C R Y P T
All your important files are encrypted.
At the moment, the cost of private key for decrypting your files is 1.5 BTC ~= 415 USD.
Your Bitcoin address for payment: 1LvjW9wyajpsC3j9RitZDip6cDcZ7jjMG5
PURCHASE PRIVATE KEY WITH BITCOIN
You can also make a payment with PaySafeCard or Ukash
In case of payment with PaySafeCard or Ukash your total payment is £ 400
PURCHASE PRIVATE KEY WITH PAYSAFECARD OR UKASH
Payment verification may take up to 12 hours.
Support
Message Center
Try to decrypt your file here
You can test the decryption service once for FREE.
Browse...

Teslacript

Os desenvolvedores do vírus de resgate TeslaCrypt divulgaram uma "chave mestra" capaz de decifrar arquivos codificados pelos vírus, permitindo abrir arquivos encriptados nas versões 3 e 4 do TeslaCrypt sem que seja necessário pagar o "resgate" exigido

Cyber Ataques

- Diferentes Tipos:
  - DDoS, worms, trojan, Malware, ...



Cyber Ataques

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Diferentes Tipos:
  - DDoS, worms, trojan, Malware, ...

Evolução contínua

COPPE UFRJ

Guerra Eletrônica

- Guerra Eletrônica (*cyber warfare*) se refere a ações de ataque ou tentativas de provocar prejuízos que um determinado Estado/Nação/Organização Internacional realiza em visando computadores, sistemas de controle e redes de computadores de outro Estado/Nação
- **60% dos profissionais predizem que ações contra governos e companhia comerciais vão piorar**
- **71% dos profissionais** predizem que as **brechas** de segurança envolvendo informações sensíveis serão de **alto risco**

Cyber Megatrends-2018

https://www.raytheon.com/cyber/cyber_megatrends

Tendências de ataques cibernéticos

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Internet of Things** é uma porta aberta e 82% dos profissionais de segurança preveem que dispositivos IoT inseguros causarão brechas de segurança em suas organizações e 80% afirmam que podem ser catastróficas
- **Extorção cibernética** (*ransomware*) aumentarão em frequência e valores de resgate, segundo 67% dos profissionais

https://www.raytheon.com/cyber/cyber_megatrends

O contraponto

- Apenas 36% dos profissionais afirmam que os seus líderes consideram a segurança cibernética como prioridade, o que significa menos investimento em tecnologia e em pessoal especializado
- Caiu em 59% o número de profissionais que acreditam que podem proteger a sua companhia de ataques cibernéticos

Cyber Megatrends

https://www.raytheon.com/cyber/cyber_megatrends

O contraponto

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Apenas seus líderes
  conside idade, o
  que sig a e em
  pessoal
- Caiu em creditam
  que poc ues
  cibernét

https: atrends

COPPE UFRJ

Prejuízos causados pelo crime cibernético

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O crime cibernético é a maior ameaça para empresas do mundo e um dos maiores problemas da humanidade. (Ginni Rometty, CEO IBM)
- O crime cibernético dará um prejuízo de 6 trilhões de dólares em 2021(PIB brasileiro em 2016 foi de 1,8 trilhão de dólares)
- O crime cibernético é mais rentável que o comércio de todas drogas ilegais combinadas
- Cybersecurity Ventures prevê que prejuízos causados por *ransomware* crescerão a 11,5 bilhões de dólares em 2019 e que um negócio será vítima de uma ataque de *ransomware* a cada 14 segundos

COPPE UFRJ

Prejuízos causados pelo crime cibernético

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **O crime cibernético é a maior ameaça para empresas do mundo e um dos maiores problemas da humanidade. (Ginni Rometty, CEO IBM)**
- O crime cibernético dará um prejuízo de 6 trilhões de dólares em 2021(PIB brasileiro em 2016 foi de 1,8 trilhão de dólares)
- O crime cibernético é mais rentável que o comércio de todas drogas ilegais combinadas
- Cybersecurity Ventures prevê que prejuízos causados por *ransomware* crescerão a 11,5 bilhões de dólares em 2019 e que um negócio será vítima de uma ataque de *ransomware* a cada 14 segundos

COPPE UFRJ

Prejuízos causados pelo crime cibernético

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O crime cibernético é a maior ameaça para empresas do mundo e um dos maiores problemas da humanidade. (Ginni Rometty, CEO IBM)
- **O crime cibernético dará um prejuízo de 6 trilhões de dólares em 2021(PIB brasileiro em 2016 foi de 1,8 trilhão de dólares)**
- O crime cibernético é mais rentável que o comércio de todas drogas ilegais combinadas
- Cybersecurity Ventures prevê que prejuízos causados por *ransomware* crescerão a 11,5 bilhões de dólares em 2019 e que um negócio será vítima de uma ataque de *ransomware* a cada 14 segundos

COPPE UFRJ

Prejuízos causados pelo crime cibernético

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O crime cibernético é a maior ameaça para empresas do mundo e um dos maiores problemas da humanidade. (Ginni Rometty, CEO IBM)
- O crime cibernético dará um prejuízo de 6 trilhões de dólares em 2021(PIB brasileiro em 2016 foi de 1,8 trilhão de dólares)
- **O crime cibernético é mais rentável que o comércio de todas drogas ilegais combinadas**
- Cybersecurity Ventures prevê que prejuízos causados por *ransomware* crescerão a 11,5 bilhões de dólares em 2019 e que um negócio será vítima de uma ataque de *ransomware* a cada 14 segundos

COPPE UFRJ

Prejuízos causados pelo crime cibernético

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O crime cibernético é a maior ameaça para empresas do mundo e um dos maiores problemas da humanidade. (Ginni Rometty, CEO IBM)
- O crime cibernético dará um prejuízo de 6 trilhões de dólares em 2021(PIB brasileiro em 2016 foi de 1,8 trilhão de dólares)
- O crime cibernético é mais rentável que o comércio de todas drogas ilegais combinadas
- **Cybersecurity Ventures prevê que prejuízos causados por *ransomware* crescerão a 11,5 bilhões de dólares em 2019 e que um negócio será vítima de uma ataque de *ransomware* a cada 14 segundos**

COPPE UFRJ

Prejuízos causados pelo crime cibernético

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

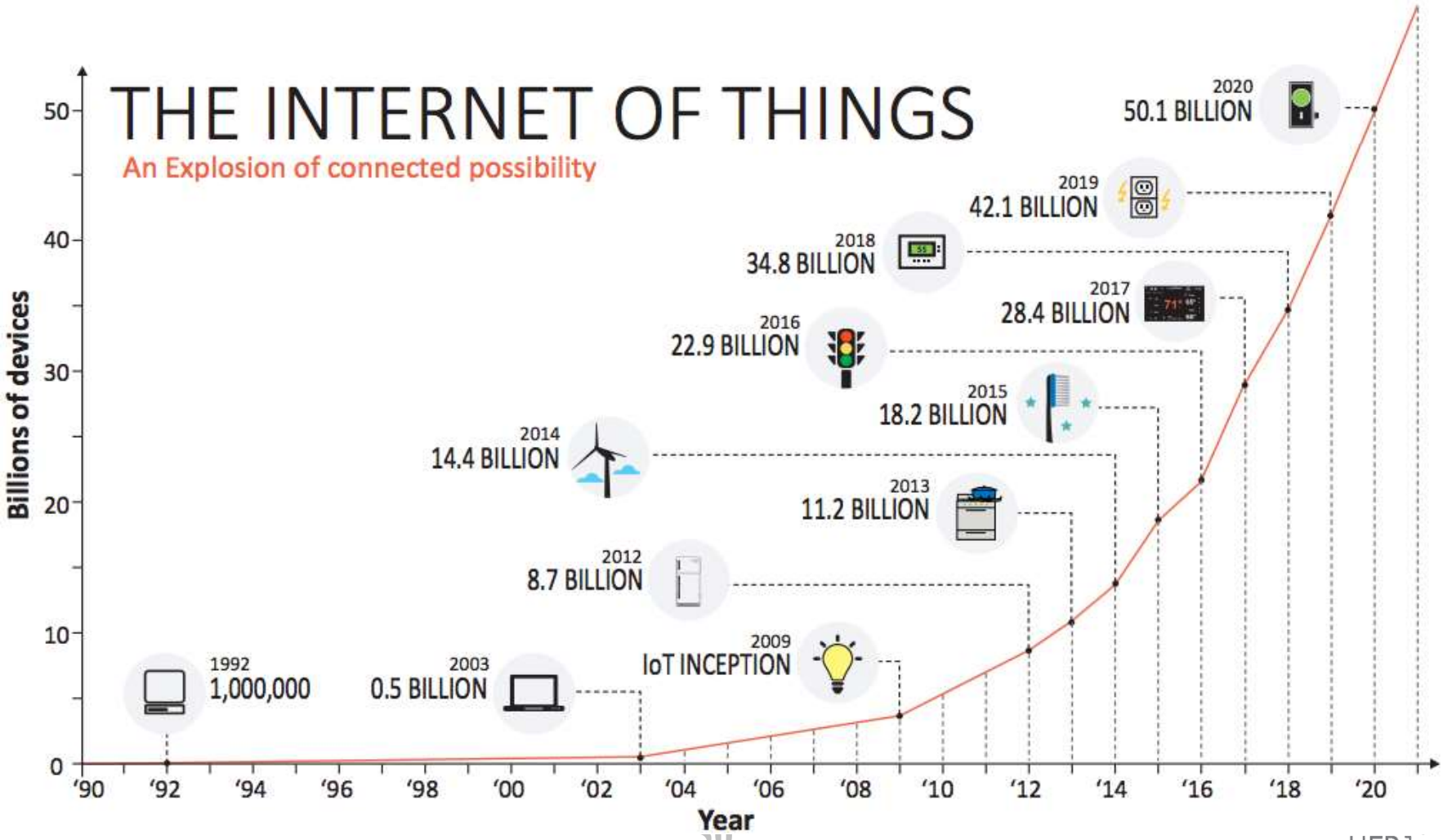
- O crime cibernético é a maior ameaça para empresas do mundo e um dos maiores problemas da humanidade. (Ginni Rometty, CEO IBM)
- O crime cibernético dará um prejuízo de 6 trilhões de dóla ão de c
- O crime cibernético é mais rentável que o comércio de todas drogas ilegais combinadas
- Cybersecurity Ventures prevê que prejuízos causados por *ransomware* crescerão a 11,5 bilhões de dólares em 2019 e que um negócio será vítima de uma ataque de *ransomware* a cada 14 segundos

Monitorar tráfegos e detectar tráfegos maliciosos é uma necessidade

COPPE UFRJ

Big Streaming Data Monitoring

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

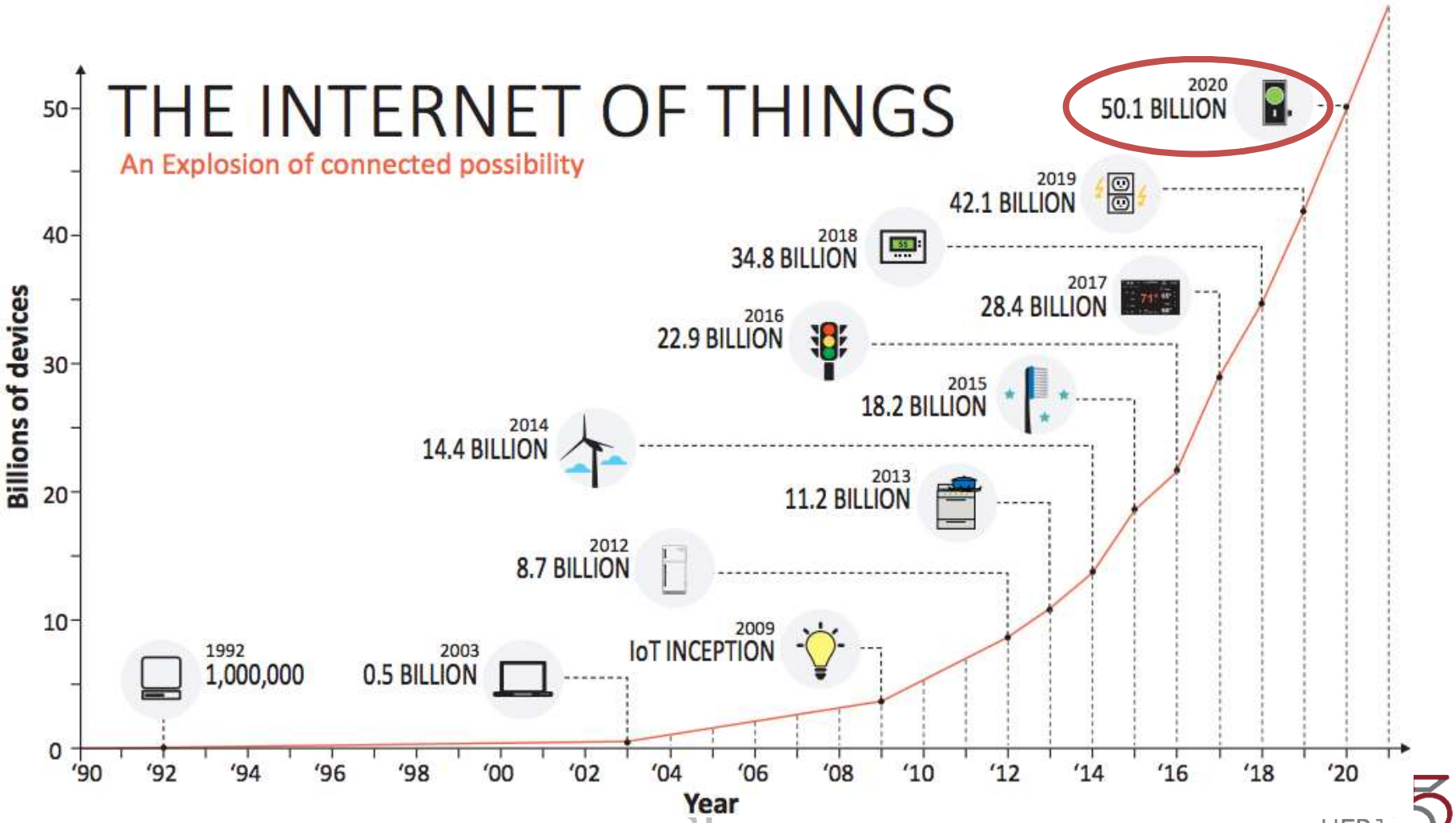


[1]https://practicalanalytics.files.wordpress.com/2012/10/newstyleofit.jpg

UFRJ

Big Streaming Data Monitoring

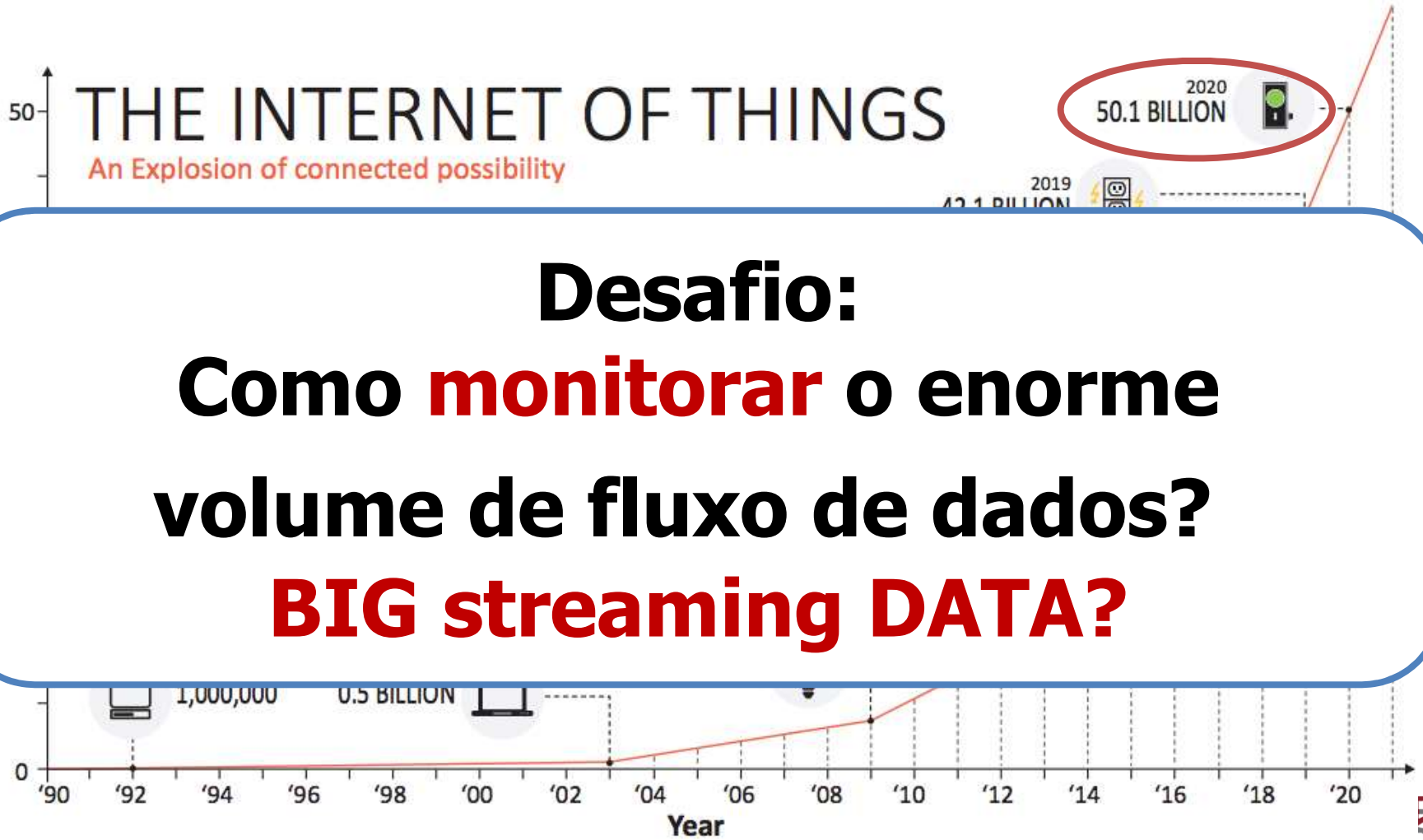
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



[1]https://practicalanalytics.files.wordpress.com/2012/10/newstyleofit.jpg

Big Streaming Data Monitoring

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Desafio:

Como monitorar o enorme volume de fluxo de dados?

BIG streaming DATA?

[1]https://practicalanalytics.files.wordpress.com/2012/10/newstyleofit.jpg

UFRJ

Os 5V de *Big Data*

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

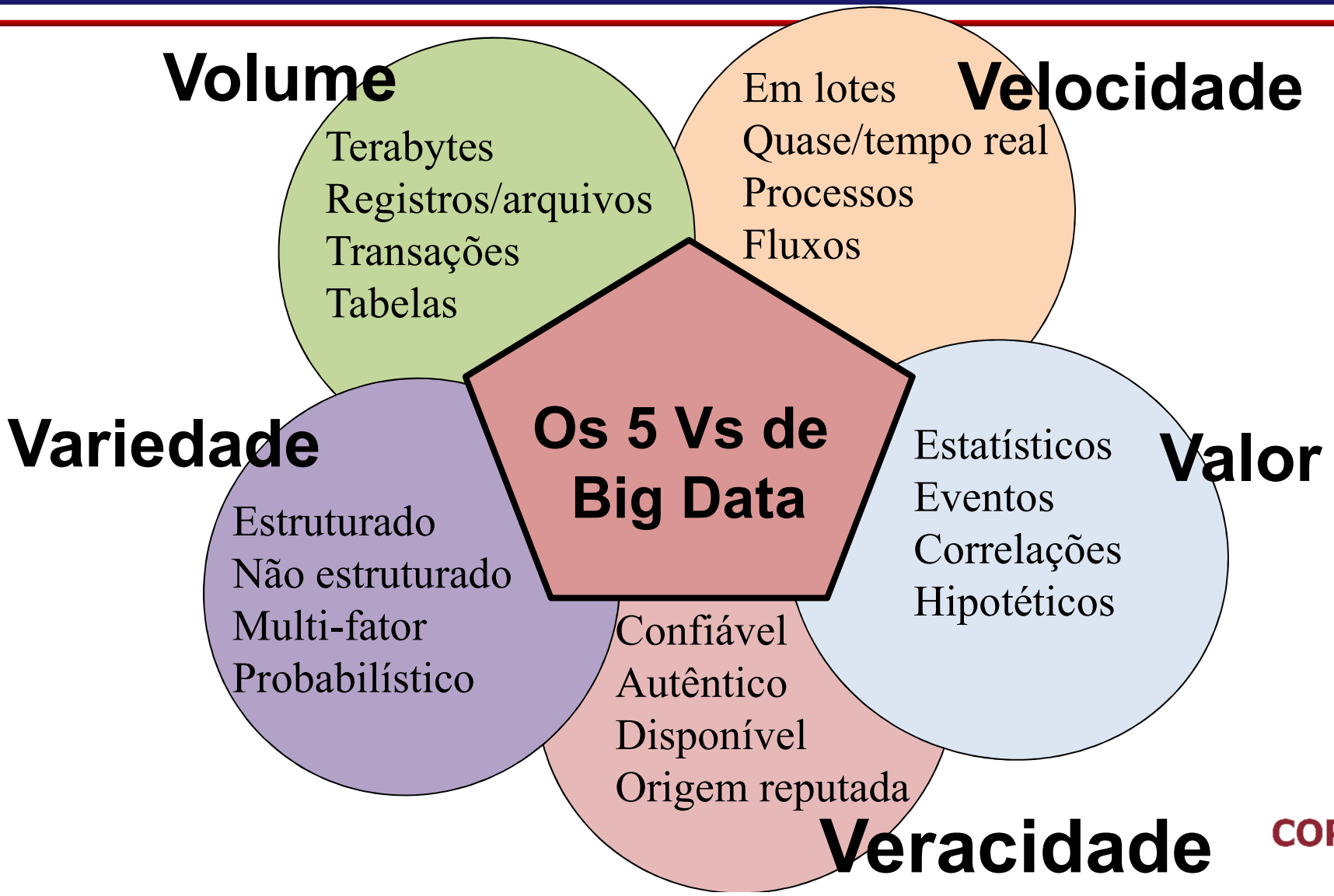
- *Big Data* se refere aos desafios tecnológicos atuais para que os sistemas de informações possam processar, armazenar e transferir a informação

- Os desafios podem ser resumidos em 5 Vs

Volume, Velocidade, Variedade, Veracidade e Valor

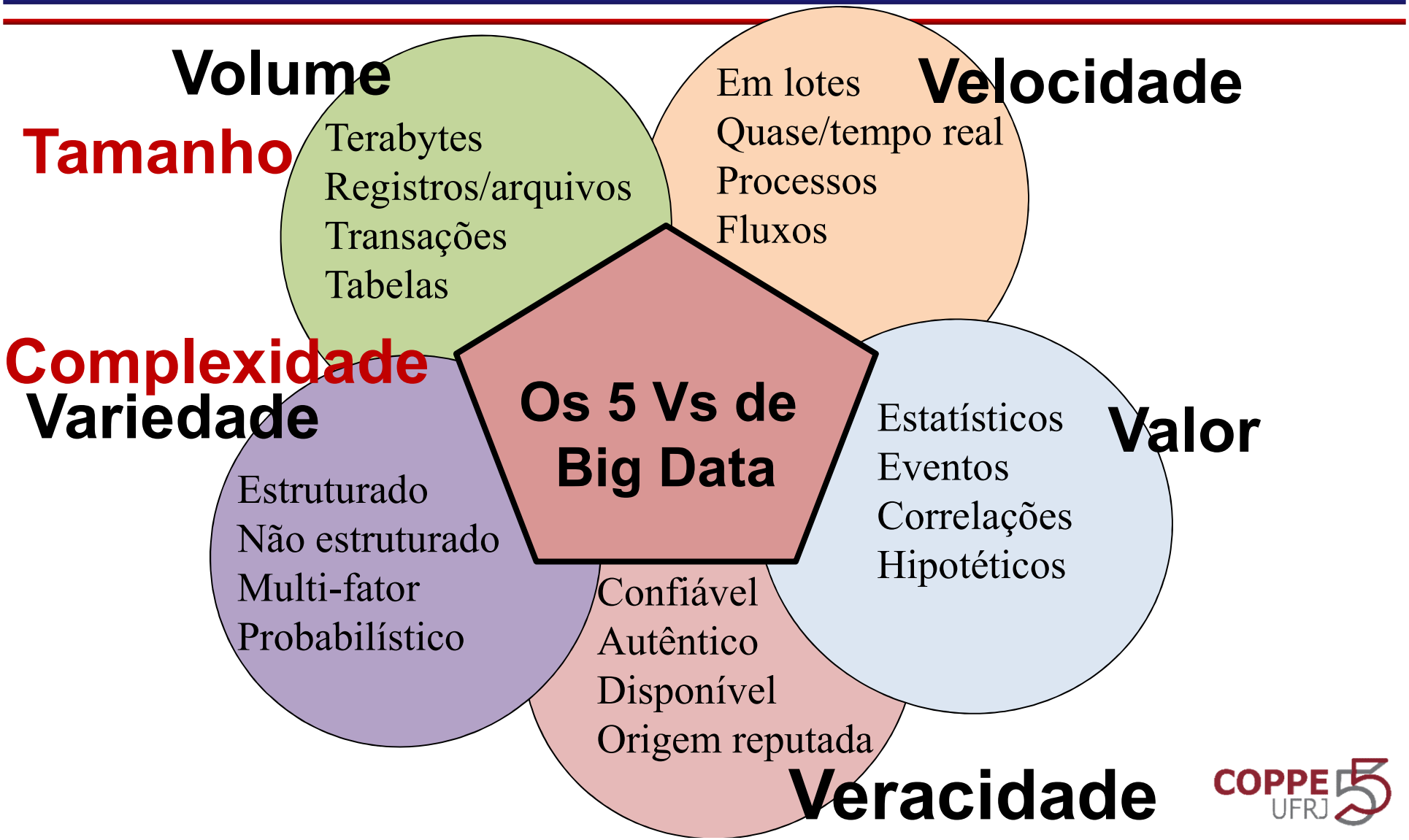
COPPE UFRJ

Os 5 Vs de Big Data



Os 5 Vs de Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Volume – Big Data

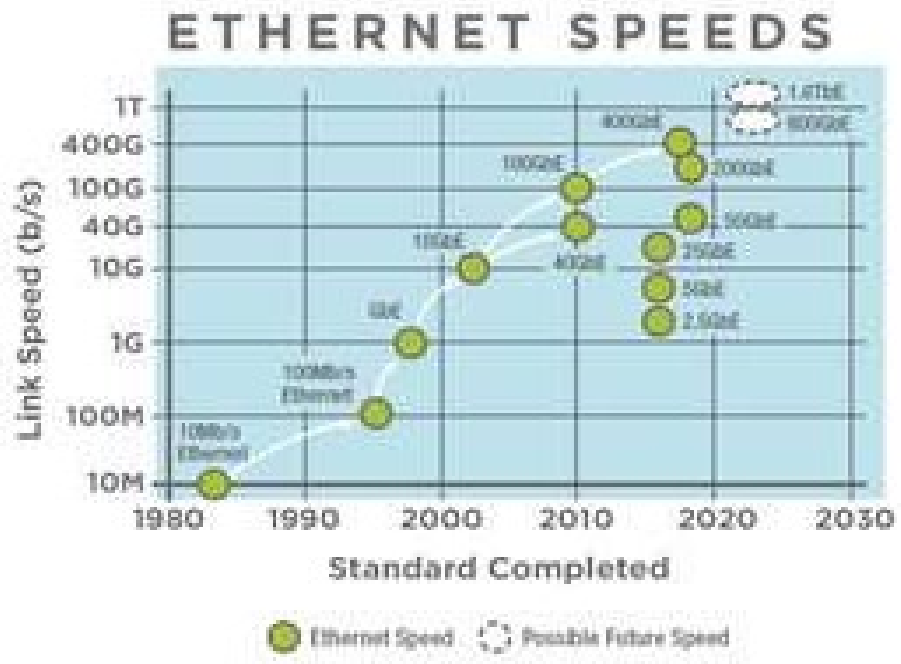
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Volume** refere-se a enorme quantidade de dados gerados a todo instante.
- As unidades medem-se em zettabytes 10^{21}, yottabytes (10^{24}) e brontobytes (10^{27})
- A **quantidade de dados gerada desde os primórdios até 2008,** em breve será **gerada a cada minuto**
- A partir de 2012, foram criados a cada dia 2,5 Exabytes (10^{18} bytes) de dados.
- A Univ. Southern Califórnia reporta que em 2007, a **humanidade enviou em difusão (TV, GPS etc.) 1.9 zettabytes de informação**
- A Univ. San Diego reporta que em 2008, os **estadunidenses consumiram 3.6 zettabytes de informação**

COPPE UFRJ

Velocidade – Big Data

Em Março de 2017, a Ethernet Alliance demonstrou **400 Gb/s** durante a *Optical Society's optical networking and communications conference and expo, OFC 2017*

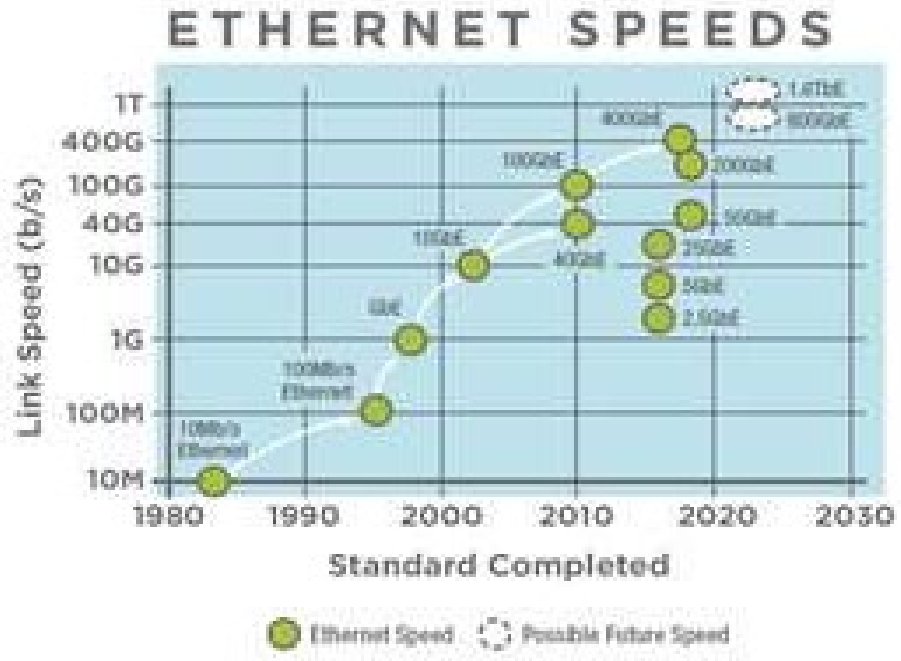


A Ethernet Alliance anunciou um plano de ação, no qual espera-se ofertar 1.6 Tb/s de Conectividade Ethernet (1.6TbE) no futuro

Velocidade – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

In March 2017, the Ethernet Alliance demonstrated **400GbE** during the Optical Society's optical networking and communications conference and expo, OFC 2017



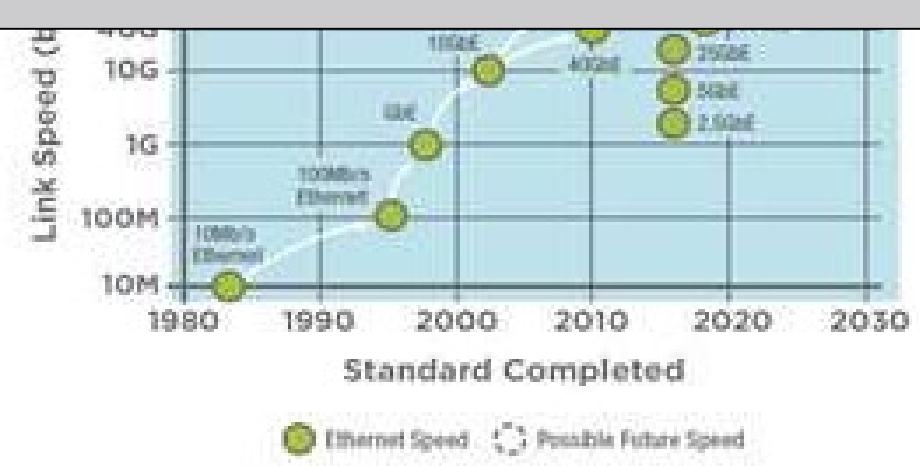
The Ethernet Alliance has announced a roadmap, in which it hopes to offer 1.6 terabit per second Ethernet (1.6TbE) connectivity in the future

COPPE UFRJ

Velocidade – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Tendência atual
25-50Gb/s Ethernet para servidores
e 100Gb/s Ethernet para rede



The Ethernet Alliance has announced a roadmap, in which it hopes to offer 1.6 terabit per second Ethernet (1.6TbE) connectivity in the future

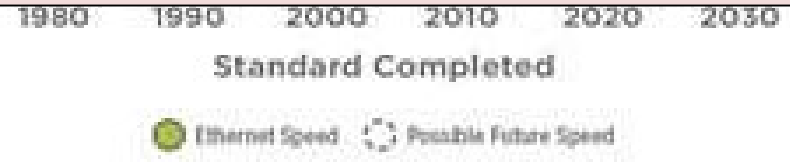
COPPE UFRJ

Velocidade – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Tendência atual
25-50Gb/s Ethernet para servidores

Como monitorar dados
a uma velocidade de 1,6 Tb/s?



The Ethernet Alliance has announced a roadmap, in which it hopes to offer 1.6 terabit per second Ethernet (1.6TbE) connectivity in the future

COPPE UFRJ

Velocidade – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Tendência atual
25-50Gb/s Ethernet para servidores

Como monitorar dados a uma velocidade de 1,6 Tb/s?

Consegue-se monitorar, detectar e agir a partir de uma mensagem que viraliza na Internet em segundos?

Variedade – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Variedade** refere-se a enorme diversidade de tipos de dados que podem ser interpretados de maneira diferente por diferentes frentes.
  - Dados estruturados: são armazenados em campos fixos de registros e de arquivos, em bancos de dados relacionais, em planilhas, sequenciados em tabelas;
  - Dados semi-estruturados: acompanham padrões heterogêneos, são mais difíceis de serem identificados, pois podem seguir diversos padrões;
  - Dados não estruturados: são uma mistura de dados com fontes diversificadas como imagens, áudios e documentos em linha (*online*).

Variedade – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

•Até **90%** de todos os dados no mundo estão na forma **de dados não estruturados** (ICD, 2011)

• Empresas que conseguem **captar a variedade** de fontes **agregam mais valor ao negócio**

• Objetivo é **a unificação da variedade de dados** em e-mails, mensagens, conversações, fotografias, áudios, sensores, telefones e cartões de crédito obtidos de *Facebook*, *Twitter* e outras redes sociais

Veracidade – Big Data

- **Veracidade** refere-se a variação que os dados capturados podem sofrer dependendo do instante de tempo ou do local no qual os dados são capturados
- **Confiança, reputação e integridade**
- **Origem, autenticidade e identificação**
  - Identificação do dado e da origem
  - Origem: domínio e autor
  - Proveniência do dado
- **Disponibilidade**
  - Pontualidade
  - Mobilidade (acesso remoto por *roaming* e federação)
- **Responsabilidade (accountability)**
  - Ação preventiva para garantir veracidade

Valor – Big Data

- **Valor** refere-se ao retorno nos resultados e intuições a serem atingidos uma vez que existe o investimento na infraestrutura requerida para coletar e analisar os dados

- Para monitoramento e detecção de ameaças é importante estimar o prejuízo financeiro e de imagem que um ataque pode causar em uma empresa, o risco, a resiliência, a acurácia na detecção da ameaças, o custo em pessoal qualificado para depurar as ameaças etc.

Valor – Big Data

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Valor** refere-se ao retorno nos resultados e intuições a serem atingidos uma vez que existe o investimento na infraestrutura requerida para coletar e analisar os dados

- **Para monitoramento e detecção de ameaças é importante estimar o prejuízo financeiro e de imagem que um ataque pode causar em uma empresa, o risco, a resiliência, a acurácia na detecção da ameaças, o custo em pessoal qualificado para depurar as ameaças etc.**

COPPE UFRJ

Sexto e Sétimo "V"

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Variabilidade (dinamicidade dos dados)**
  - Propriedade que reflete a mudança do dados durante seu processamento ou ciclo de vida

- **Vulnerabilidade**
  - Propriedade que indica maiores riscos de brechas de segurança e ataques mais efetivos devido a quantidade de zumbis

COPPE UFRJ

Aplicações de Processamento de Fluxo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Operações no mercado financeiro requerem processamento de dezenas de milhares de mensagens por segundo e latências menores que um segundo
- Detecção de fraudes em tempo real de serviços financeiros e telefones celulares requerem respostas da ordem do segundo.
- processos de automação industrial
- Aplicações militares de processamento de sinalizações de atividade e GPS
- Monitoramento de sensores em cidades inteligentes
- **Monitoramento de tráfego da Internet para detecção de intrusões**

Base de dados Convencionais vs Fluxos

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Sistemas de Gerência de Base de Dados (DBMS)	**X**	**Sistemas de Gerência de Fluxo de Dados (DSMS)**
1.Dados persistentes		1.Fluxo de dados voláteis
2.Acesso aleatório		2.Acesso sequencial
3.Requisições uma por vez		3.Requisições contínuas
4.Armazenamento secundário ilimitado		4.Memória principal limitada
5.Apenas o estado atual é relevante		5.Ordem da entrada importa
6.Frequência de atualização baixa		6.Frequência de atualização muito alta
7.Baixa restrição de tempo		7.Necessidade de tempo real
8.Assume dados exatos		8.Assume dados incorretos e desatualizados
9.Processamento de requisições planificável		9.Características e chegada de dados variáveis

Plataformas de Processamento de Fluxo de **Primeira Geração**

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A primeira geração se baseia em sistemas de banco de dados para **base de dados "ativas"**
  - Avaliam as regras expressas como pares condição-ação quando novos eventos chegam
  - sistemas eram limitados e não comportavam fluxos com grandes volumes de dados
- Ingres (INteractive Graphics and REtrieval System) and Postgres (POSTinGRES), propos um modelo de base de dados relacional orientada a objeto, permitiu análise de séries temporais, Michael Stonebreaker, Universidade de Berkeley e depois MIT, 1991
- Starburst Active Database Rule System – Jennifer Widom, Universidade de Stanford, 1992
- NiagaraCQ – sistema de requisição contínua escalável, Chen et al., Universidade de Wiconsin-Madison, 2000

COPPE UFRJ

Plataformas de Processamento de Fluxo de **Primeira Geração**

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A primeira geração se baseia em sistemas de banco de dados para **base de dados "ativas"**

A **Apama Stream Analytics,** fundada em 1999, foi primeira empresa de análise de eventos em tempo real, foi vendida em 2005 para a Progress Software Coorporation por 25 milhões de dólares, permitia monitorar eventos, analisá-los e realizar ações em milissegundos.

- al., Universidade de Wiconsin-Madison, 2000

COPPE UFRJ

Plataformas de Processamento de Fluxo de **Segunda Geração**

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A segunda geração foca em **estender a linguagem de consulta estruturada (*Structured Query Language* - SQL) para processar fluxos, explorando as semelhanças entre um fluxo e uma consulta (*query*)**
- Projeto **STREAM**: STanford stREam datA Manager, o primeiro sistema de gerenciamento de fluxo de dados de uso geral (*Data Stream Management System* - DSMS), Universidade de Stanford, 2003
- **TelegraphCQ:** linguagem para executar continuamente consultas SQL com base no banco de dados Postgres, Universidade de Berkeley, em 2003
- Projeto **AURORA** propõe um modelo e uma arquitetura para aplicações de monitoramento de fluxo de dados, Universidades de Brandeis e Brown e M.I.T., 2003
- Projeto **Cayuga**: um sistema escalável para processamento de fluxo de dados, produz alta vazão até milhares de eventos por segundo, sistema *publish/subscribe*, escala tanto p/ taxa de chegada de eventos quanto p/ o número de consultas, Universidade de Cornell, 2007

Plataformas de Processamento de Fluxo de **Segunda Geração**

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A segunda geração foca em **estender a linguagem de consulta estruturada (*Structured Query Language* - SQL) para processar**

Aurora/Borealis impulsionaram em 2003
a fundação da empresa StreamBase System, que lançou comercialmente a plataforma StreamBase para processamento de eventos complexos
(Complex Event Processing - CEP),
vendida para a empresa TIBCO Software em 2013.

TelegraphCQ deu origem em 2009 a empresa da Truviso Analytic e em 2012 foi adquirida pela Cisco.

publish/subscribe, escala tanto p/ taxa de chegada de eventos quanto p/ o número de consultas, Universidade de Cornell, 2007

Plataformas de Processamento de Fluxo de **Terceira Geração**

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A terceira geração foca o **processamento distribuído escalável** de fluxos de dados em aglomerados computacionais
- Objetiva atender à necessidade das empresas associadas à Internet de processar grandes volumes de dados produzidos à alta velocidade
- Google revoluciona o processamento distribuído propondo o modelo de programação MapReduce para o processamento paralelo escalável de grandes volumes de dados em aglomerados.
  - A ideia chave de **espalhar-processar-combinar** é usada para realizar de forma escalável diferentes tarefas em paralelo em servidores de um aglomerado
- A plataforma Hadoop de código aberto democratiza o acesso a analíticas em grandes massas de dados.

Plataformas de Processamento de Fluxo de **Terceira Geração**

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A terceira geração foca o **processamento distribuído escalável** de fluxos de dados em aglomerados computacionais
- Objetiva atender à necessidade das empresas associadas à

As plataformas de código aberto Apache **Storm**, Apache **Spark** e Apache **Flink** são as principais plataformas de processamento distribuído de fluxo em aglomerados

realizar de forma escalável diferentes tarefas em paralelo em servidores de um aglomerado

- A plataforma Hadoop de código aberto democratiza o acesso a analíticas em grandes massas de dados.

COPPE UFRJ

Sumário

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Parte I**
  - Introdução e Conceitos Fundamentais de Processamento de Fluxo
- **Parte II**
  - Plataformas de Código Aberto para Processamento Distribuído de Fluxo
- **Parte III**
  - Detecção de Ameaças usando Aprendizado de Máquina
- **Parte IV**
  - Detecção de Ameaças Inéditas em Tempo Real a partir de Treinamento Adaptativo
- **Parte V**
  - Considerações Finais e Problemas em Aberto
- **Aula Prática**

COPPE UFRJ

PARTE II

Plataformas de Código Aberto para Processamento Distribuído de Fluxo

APACHE HADOOP

Apache Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **2004**
  - Google File System (GFS)
  - Google MapReduce
- **2006**
  - Início do projeto Hadoop – Doug Cutting (Stanford)
  - Yahoo - maior contribuidor
- **2010**
  - Versão 1.0 do Hadoop
- Origem do nome Hadoop?

COPPE UFRJ

Apache Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **2004**
  - Google File System (GFS)
  - Google MapReduce
- **2006**
  - Início do projeto Hadoop – Doug Cutting (Stanford)
  - Yahoo - maior contribuidor
- **2010**
  - Versão 1.0 do Hadoop
- Origem do nome Hadoop?
  - Nome do elefante de pelúcia amarelo do seu filho

"The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid's term."

COPPE UFRJ

Distribuições do Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Código aberto
  - Apache Hadoop
- Comerciais
  - Cloudera
  - Hortonworks
  - MapR
  - Amazon Web Services (AWS)
  - Windows Azure HDInsight

Empresas que usam Hadoop

- Facebook
- Yahoo
- Amazon
- EBay
- American Airlines
- The New York Times
- Federal Reserve Board
- IBM
- Orbitz

Exemplos de Aplicações de Big Data em Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A9.com – Amazon: para construção dos índice de busca de produtos da Amazon, processa diariamente milhões de sessões para analíticas, usando JAVA e API streaming, o aglomerado varia de **1 a 100 nós**
- Yahoo! : mais de **100,000 CPUs em ~20,000** computadores; o maior aglomerado é de **2000 nós** (2*4CPU boxes com 4TB de HD cada); usados para busca Web
- AOL : aglomerado com 50 computadores, Intel Xeon, dual processors, dual core, cada um com 6GB RAM e 800 GB de HD, dando um total de **37 TB de capacidade**, usado para diversas coisas de geração de estatísticas a execução de algoritmos avançados para análise comportamental
- Facebook: para armazenar cópias de logs internos e fontes de dados, e usá-los como fonte para relatórios e analíticas e aprendizado de máquina; **aglomerado com 320 computadores, com 2,560 núcleos e por volta de 1.3 PB armazenamento cru (raw)**

Vantagens do Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Escalabilidade** - pode armazenar e processar petabytes de forma confiável
- **Tolerância a Falhas** - automaticamente redistribui as tarefas computacionais quando há falhas
- **Econômico** - distribui processamento em máquinas de baixo custo
- **Eficiente** - processa em paralelo nos nós onde os dados estão localizados

Hipóteses de Projeto do Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Escalabilidade → aglomerado com **grande número** de computadores
- Baixo custo → **hardware** do aglomerado irá **falhar**
- A **vazão é mais importante** que latência
- Arquivos de GB a TB de tamanho
  - Modelo de acesso das aplicações **write-once-read-many**
  - **Mover processamento** é mais barato que mover dados

Hadoop - Principais Componentes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

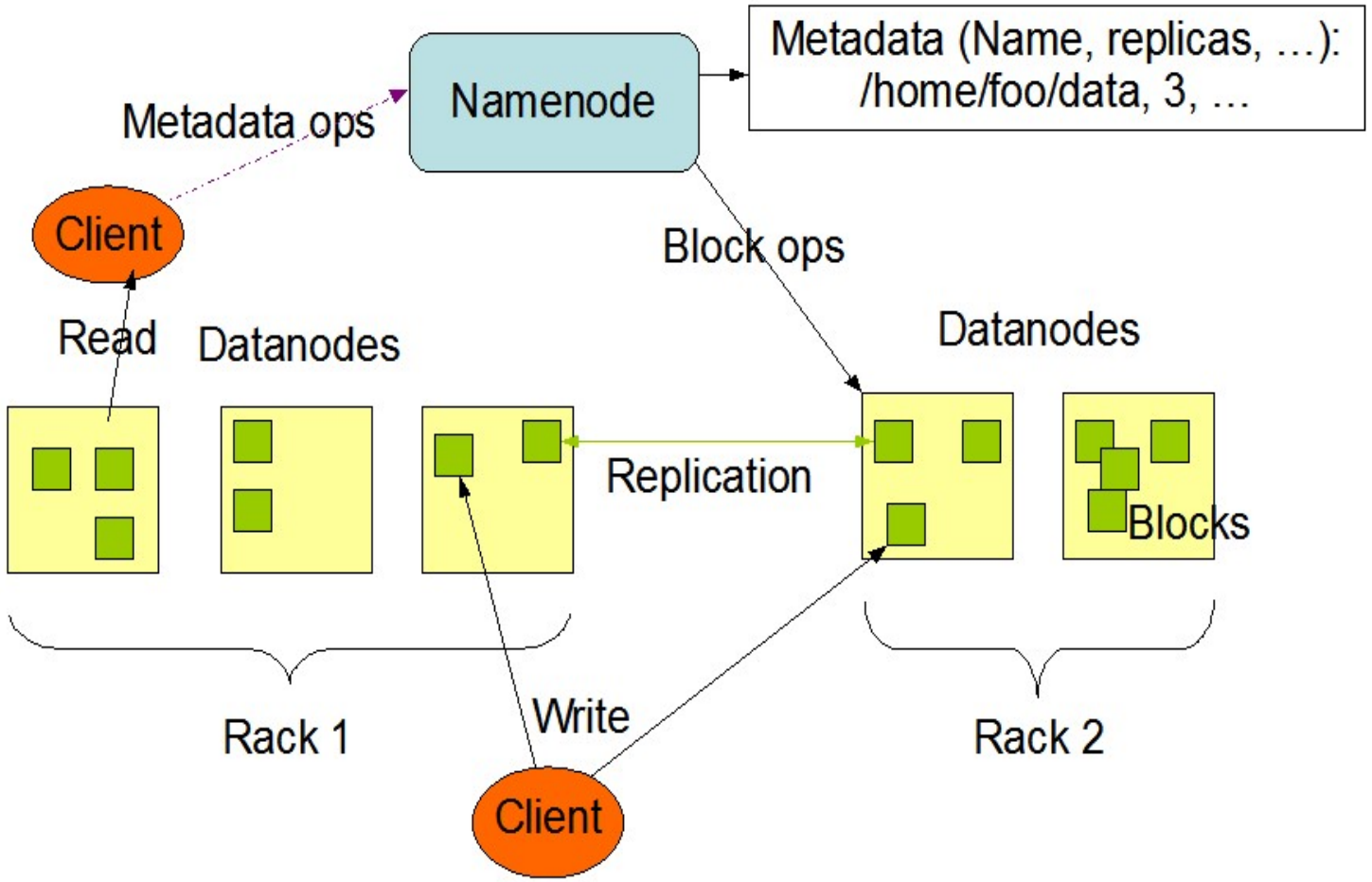
- **Hadoop Distributed File System (HDFS)**
  - Sistema distribuído de arquivos através de blocos
  - Tamanho padrão de um bloco: 64 MB
  - Todo bloco é replicado 3 vezes por padrão
  - Tolerante a falhas
    - Sinalização de atividade (*heartbeat*) a cada 3 segundos
- **MapReduce**
  - Modelo de programação proposto pelo Google para facilitar o processamento de grandes volumes de dados
  - Motor de computação em tempo diferenciado (offline)
  - Duas operações básicas: ***map*** e ***reduce***

COPPE UFRJ

Hadoop Distributed File System (HDFS)

- Sistema de arquivos distribuído projetado para ser implantado em **hardware de baixo custo**
  - Altamente **tolerante a falhas**
    - Replicação de dados
  - Adequado para aplicativos que possuem grandes conjuntos de dados
  - Fornece **acesso de alto desempenho** aos dados do aplicativo
- Parte do projeto Apache Hadoop Core (http://hadoop.apache.org/core/)

Arquitetura HDFS



HDFS - Namenode

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Elemento central do sistema de arquivos HDFS
  - Mantém a **árvore de diretórios** dos arquivos do sistema
  - **Não armazena dados**, apenas controla a posição dos arquivos nos outros nós
  - Contém metadados dos arquivos do sistema
    - Informações de replicação

- Caso sofra algum problema o sistema todo não funciona
  - **Ponto único de falha**

HDFS - Datanode

- **Armazena os dados** do HDFS
- Podem existir muitos Datanodes
- As réplicas são armazenadas no Datanodes

- Caso um Datanode sofra algum problema o aglomerado não é afetado
  - Datanodes enviam mensagens periódicas ao Namenode
  - Uma falha de Datanode é identificada pela ausência de mensagens
    - Pode-se buscar as réplicas em outros Datanodes

O Modelo de Programação MapReduce

- Desenvolvido pela Google
- Computação baseada na **distribuição** e **coleta** de dados
- Programação de estilo **funcional** → naturalmente paralelizável em aglomerados
- Sistema subjacente (razão do sucesso do Hadoop)
  - Provê o particionamento dos dados de entrada
  - Distribui nativamente a execução do programa em várias máquinas
  - Lida com falhas de máquinas
  - Gerencia a comunicação entre máquinas

Map Reduce

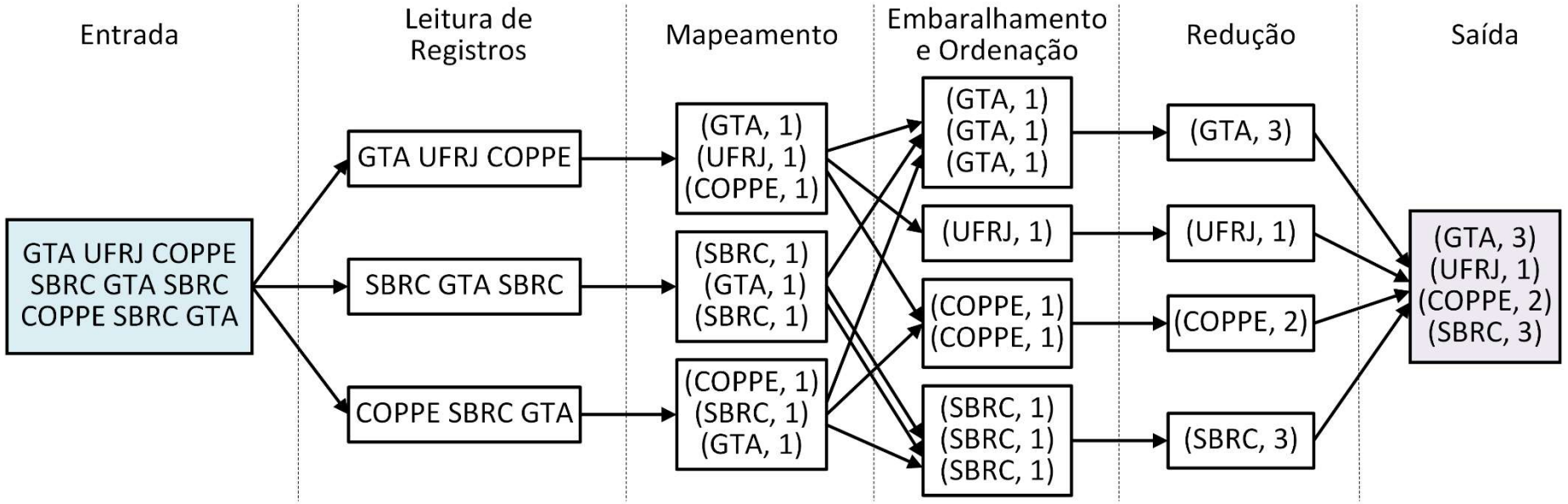
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O sistema particiona a entrada e a fornece para diferentes instâncias de Mapeamento
- **Map** (chave, valor) → (chave ', valor')
- O sistema coleta os pares (chave', valor') e os distribui para várias funções redutoras de modo que cada função obtenha os pares com a mesma chave’
- Cada redutor produz uma única saída de arquivo
- **Map** e **Reduce** são funções escritas pelo usuário

COPPE UFRJ

MapReduce - Exemplo de contagem de palavras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

MapReduce - Exemplo de contagem de palavras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

```
1   map(String input_key, String input_value):
2         // input_key: nome do documento
3         // input_value: conteúdo do documento
4         for each word w in input_value:
5               EmitIntermediate(w, "1");
6   reduce(String output_key, Iterator intermediate_values):
7        // output_key: a word
8        // output_values: a list of counts
9        int result = 0;
10       for each v in intermediate_values:
11            result += ParseInt(v);
12            Emit(AsString(result));
```


COPPE UFRJ

MapReduce - Exemplo de contagem de palavras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

```
1   map(String input_key, String input_value):
2       // input_key: nome do documento
3       // input_value: conteúdo do documento
4       for each word w in input_value:
5           EmitIntermediate(w, "1");
6   reduce(String output_key, Iterator intermediate_values):
7       // output_key: a word
8       // output_values: a list of counts
9       int result = 0;
10      for each v in intermediate_values:
11          result += ParseInt(v);
12          Emit(AsString(result));
```

A função *map* emite o valor '1' associado à chave para cara palavra do documento de entrada

COPPE UFRJ

MapReduce - Exemplo de contagem de palavras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

```
1   map(String input_key, String input_value):
2       // input_key: nome do documento
3       // input_value: conteúdo do documento
4       for each word w in input_value:
5           EmitIntermediate(w, "1");
6   reduce(String output_key, Iterator intermediate_values):
7       // output_key: a word
8       // output_values: a list of counts
9       int result = 0;
10      for each v in intermediate_values:
11          result += ParseInt(v);
12          Emit(AsString(result));
```

A função *map* emite o valor '1' associado à chave para cara palavra do documento de entrada

A função de *reduce* soma todas as contagens emitidas para uma mesma chave, ou seja, uma mesma palavra

COPPE UFRJ

Tolerância a Falhas do MapReduce

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Falha do trabalhador**
  - O mestre checa cada trabalhador periodicamente
  - Se não houver reposta o mestre marca trabalhador como falho
  - Tarefas de mapeamento e redução no trabalhador falho são redefinidas como inativas e, portanto, tornam-se elegíveis para **agendamento em outros trabalhadores**
- **Falha do Mestre**
  - O mestre grava **pontos de verificação** periódicos das estruturas de dados
  - Se a tarefa mestre falha, uma nova cópia poderá ser iniciada a partir do **último estado** de ponto de verificação. No entanto, na maioria dos casos, **o usuário reinicia o trabalho**

COPPE UFRJ

A execução no Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

1. A biblioteca MapReduce divide os arquivos do programa do usuário em M pedaços de 16 MB ou 64 MB e inicia muitas cópias do programa nas máquinas do aglomerado
2. Uma cópia é Mestre e as outras são trabalhadores atribuídos pelo Mestre. O Mestre seleciona trabalhadores ociosos para atribuir M tarefas de Map e R tarefas de Reduce
3. Um trabalhador com tarefa Map analisa os valores chave/valor a passa para a função Map definida pelo usuário. Os valores intermediários chave/valor são armazenados
4. As localizações dos pares no disco local são retornadas para o Mestre que redireciona estas localizações para os trabalhadores de Reduce

A execução no Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

1. A biblioteca MapReduce divide os arquivos do programa do usuário em M pedaços de 16 MB ou 64 MB e inicia muitas cópias do programa nas máquinas do aglomerado
2. Uma cópia é Mestre e as outras são trabalhadores atribuídos pelo Mestre. O Mestre seleciona trabalhadores ociosos para atribuir M tarefas de Map e R tarefas de Reduce

1
MapReduce divide os dados em M pedaços de 16 MB ou 64 MB e faz cópias do programa no aglomerado

A execução no Hadoop

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

1. A biblioteca MapReduce divide os arquivos do programa do usuário em M pedaços de 16 MB ou 64 MB e inicia muitas cópias do programa nas máquinas do aglomerado
2. Uma cópia é Mestre e as outras são trabalhadores atribuídos pelo Mestre. O Mestre seleciona trabalhadores ociosos para atribuir M tarefas de Map e R tarefas de Reduce
3
4

2
Uma cópia é o mestre
e as outras são escravos
M tarefas Map e R tarefas Reduce
Mestre seleciona trabalhadores ociosos

3
Tarefa Map analisa os pares chave/valor
Executa a função Map do usuário
Armazena valores intermediários

atribuir M tarefas de Map e R tarefas de Reduce

3. Um trabalhador com tarefa Map analisa os valores chave/valor a passa para a função Map definida pelo usuário. Os valores intermediários chave/valor são armazenados
4. As localizações dos pares no disco local são retornadas para o Mestre que redireciona estas localizações para os trabalhadores de Reduce

A execução no Hadoop

1

2

3

4

As localizações dos pares no disco são enviadas para o mestre que redireciona para as tarefas Reduce

intermediários chave/valor são armazenados

4. As localizações dos pares no disco local são retornadas para o Mestre que redireciona estas localizações para os trabalhadores de Reduce

A execução no Hadoop (cont.)

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

5. O trabalhador Reduce faz chamadas de procedimentos remotas para obter os dados armazenados no disco local dos trabalhadores. O trabalhador Reduce ordena e grupa todas as ocorrências de uma mesma chave
6. O trabalhador Reduce itera sobre os dados intermediários ordenados e para cada chave única intermediária encontrada, ele passa a chave e o correspondente conjunto de valores intermediários para a função Reduce do usuário. A saída da função Reduce é anexada ao arquivo de saída final para esta partição de Reduce
7. Ao finalizar todas as tarefas Map e Reduce o Master avisa o usuário e a saída da execução está disponível nos arquivos de saída R

A execução no Hadoop (cont.)

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

5. O trabalhador Reduce faz chamadas de procedimentos remotas para obter os dados armazenados no disco local dos trabalhadores. O trabalhador Reduce ordena e grupa todas as ocorrências de uma mesma chave

5
Tarefa Reduce faz RPCs p/ dados armazenados - disco local dos trabalhadores
A tarefa Reduce ordena e agrupa ocorrências de uma mesma chave

arquivos de saída R

COPPE UFRJ

A execução no Hadoop (cont.)

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

6

A tarefa Reduce itera sobre os dados intermediários ordenados

Para cada chave única intermediária, passa a chave e os valores intermediários para a função Reduce

A saída da função Reduce é anexada ao arquivo de saída final desta partição de Reduce

A execução no Hadoop (cont.)

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

7
O mestre avisa ao usuário
o fim das tarefas de Map e Reduce
A saída da execução está disponível
nos arquivos de saída

7. Ao finalizar todas as tarefas Map e Reduce o Master avisa o usuário e a saída da execução está disponível nos arquivos de saída R

COPPE UFRJ

APACHE SPARK

Apache Spark - Histórico

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **2011**
  - UC Berkeley recebe US$ 30M para pesquisa em análise de dados em tempo real
  - AmpLab é criado
- **2012**
  - AmpLab desenvolve o Spark
  - Resilient Distributed Datasets (RDD)
- **2013**
  - Spark é doado para a Fundação Apache
  - Spark Streaming é desenvolvido
  - Desenvolvedores do Spark criam a empresa Databricks

COPPE UFRJ

Apache Spark - Motivação

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Spark substitui a função **MapReduce do Hadoop**

- Spark faz em memória o que o MapReduce do Hadoop faz em disco
  - Hadoop – acesso a disco lento para processos iterativos
  - Spark – Armazenamento temporário dos dados (*caching*)
    - Bom para aprendizado de máquina **- 10 a 100x mais rápido**
- Spark provê mais facilidades de paralelismo
  - Spark - paralelismo por thread
  - Hadoop – paralelismo por processo
- Spark provê alocação dinâmica de recursos
  - Hadoop define estaticamente os processos trabalhadores
    - Desperdiça recursos, pois os trabalhadores Map e Reduce executam apenas em as suas fases

COPPE UFRJ

Apache Spark - Motivação

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Spark substitui a função MapReduce do Hadoop

  - Spark – Armazenamento temporário dos dados (*caching*)
    - Bom para aprendizado de máquina **- 10 a 100x mais rápido**
- Spark provê mais facilidades de paralelismo
  - Spark - paralelismo por thread
  - Hadoop – paralelismo por processo
- Spark provê alocação dinâmica de recursos
  - Hadoop define estaticamente os processos trabalhadores
    - Desperdiça recursos, pois os trabalhadores Map e Reduce executam apenas em as suas fases

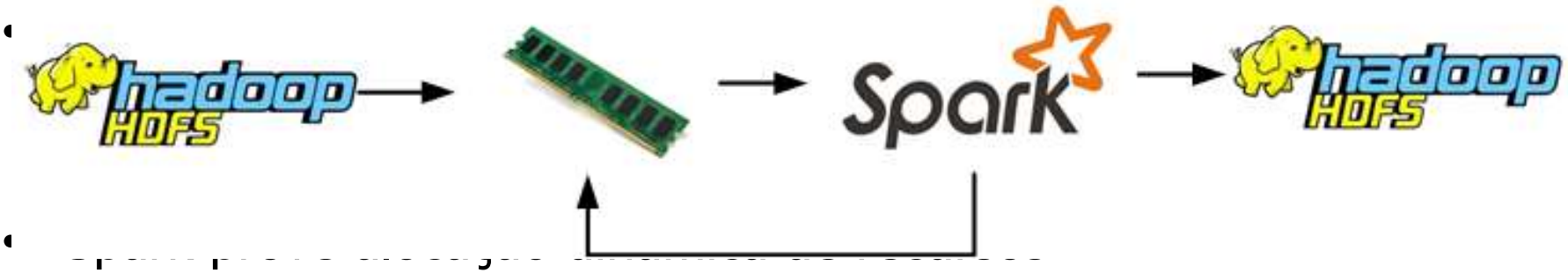
COPPE UFRJ

Apache Spark - Motivação

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Spark substitui a função **MapReduce do Hadoop**

- Spark – Armazenamento temporário dos dados (*caching*)
  - Bom para aprendizado de máquina **- 10 a 100x mais rápido**



- Hadoop define estaticamente os processos trabalhadores
  - Desperdiça recursos, pois os trabalhadores Map e Reduce executam apenas em as suas fases

Apache Spark vs Hadoop contagem de palavras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Table 3: Overall Results: Word Count

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (GB)	1	1	40	40	200	200
Number of map tasks	9	9	360	360	1800	1800
Number of reduce tasks	8	8	120	120	120	120
Job time (Sec)	30	64	70	180	232	630
Median time of map tasks (Sec)	6	34	9	40	9	40
Median time of reduce tasks (Sec)	4	4	8	15	33	50
Map Output on disk (GB)	0.03	0.015	1.15	0.7	5.8	3.5

Fonte: Shi, Juwei, et al. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8.13 (2015): 2110-2121.

COPPE UFRJ

Apache Spark vs Hadoop contagem de palavras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Table 3: Overall Results: Word Count

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (GB)	1	1	40	40	200	200
Number of map tasks	9	9	360	360	1800	1800
Number of reduce tasks	8	8	120	120	120	120
Job time (Sec)	30	64	70	180	232	630
Median time of [illegible]	[illegible]	[illegible]	[illegible]	40	9	40
Median time of [illegible]	[illegible]	[illegible]	[illegible]	15	33	50
Map Output or[illegible]	[illegible]	[illegible]	[illegible]	0.7	5.8	3.5

Spark > Hadoop

Fonte: Shi, Juwei, et al. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8.13 (2015): 2110-2121.

COPPE UFRJ

Apache Spark vs Hadoop
Ordenação

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Table 5: Overall Results: Sort

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (GB)	1	1	100	100	500	500
Number of map tasks	9	9	745	745	4000	4000
Number of reduce tasks	8	8	248	60	2000	60
Job time	32s	35s	4.8m	3.3m	44m	24m
Sampling stage time	3s	1s	1.1m	1s	5.2m	1s
Map stage time	7s	11s	1.0m	2.5m	12m	13.9m
Reduce stage time	11s	24s	2.5m	45s	26m	9.2m
Map output on disk (GB)	0.63	0.44	62.9	41.3	317.0	227.2

Fonte: Shi, Juwei, et al. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8.13 (2015): 2110-2121.

COPPE UFRJ

Apache Spark vs Hadoop
Ordenação

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Table 5: Overall Results: Sort

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (GB)	1	1	100	100	500	500
Number of map tasks	9	9	745	745	4000	4000
Number of reduce tasks	8	8	248	60	2000	60
Job time	32s	35s	4.8m	3.3m	44m	24m
Sampling stage time	3s	1s	1.1m	1s	5.2m	1s
Map stage time					2m	13.9m
Reduce stage time					6m	9.2m
Map output on disk (C					7.0	227.2

Hadoop > Spark

Fonte: Shi, Juwei, et al. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8.13 (2015): 2110-2121.

COPPE UFRJ

Apache Spark vs Hadoop
Algoritmos iterativos

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Table 7: Overall Results: K-Means

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (million records)	1	1	200	200	1000	1000
Iteration time 1st	13s	20s	1.6m	2.3m	8.4m	9.4m
Iteration time Subseq.	3s	20s	26s	2.3m	2.1m	10.6m
Median map task time 1st	11s	19s	15s	46s	15s	46s
Median reduce task time 1st	1s	1s	1s	1s	8s	1s
Median map task time Subseq.	2s	19s	4s	46s	4s	50s
Median reduce task time Subseq.	1s	1s	1s	1s	3s	1s
Cached input data (GB)	0.2	-	41.0	-	204.9	-

Fonte: Shi, Juwei, et al. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8.13 (2015): 2110-2121.

COPPE UFRJ

Apache Spark vs Hadoop
Algoritmos iterativos

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Table 7: Overall Results: K-Means

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (million records)	1	1	200	200	1000	1000
Iteration time 1st	13s	20s	1.6m	2.3m	8.4m	9.4m
Iteration time Subseq.	3s	20s	26s	2.3m	2.1m	10.6m
Median map task time 1st	11s	19s	15s	46s	15s	46s
Median reduce task time 1st	1s	1s	1s	1s	8s	1s
Median map task					4s	50s
Median reduce ta					3s	1s
Cached input data					204.9	-

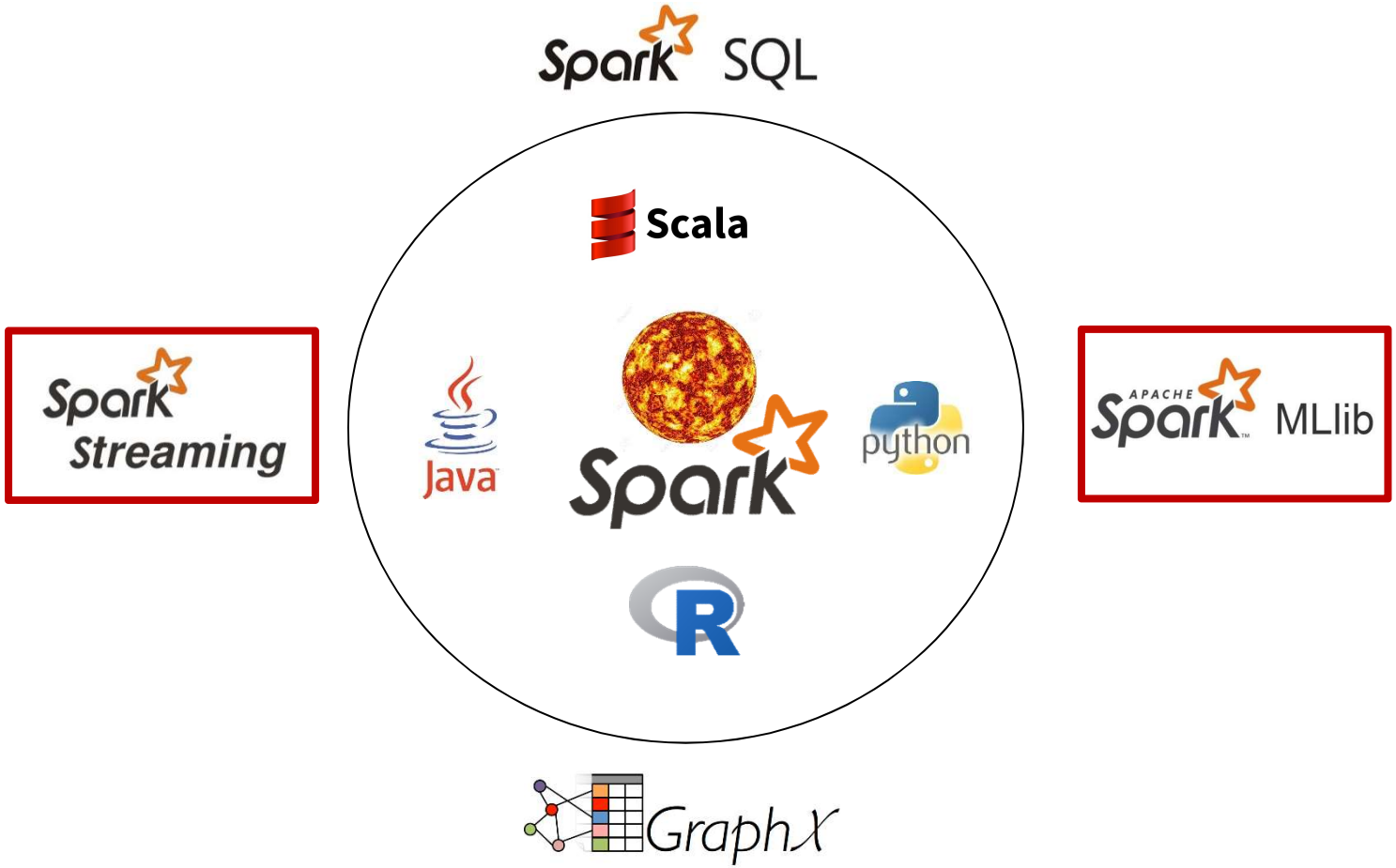
Spark >> Hadoop

Fonte: Shi, Juwei, et al. "Clash of the titans: Mapreduce vs. spark for large scale data analytics." *Proceedings of the VLDB Endowment* 8.13 (2015): 2110-2121.

COPPE UFRJ

Apache Spark - Ecossistema

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Apache Spark - Arquitetura

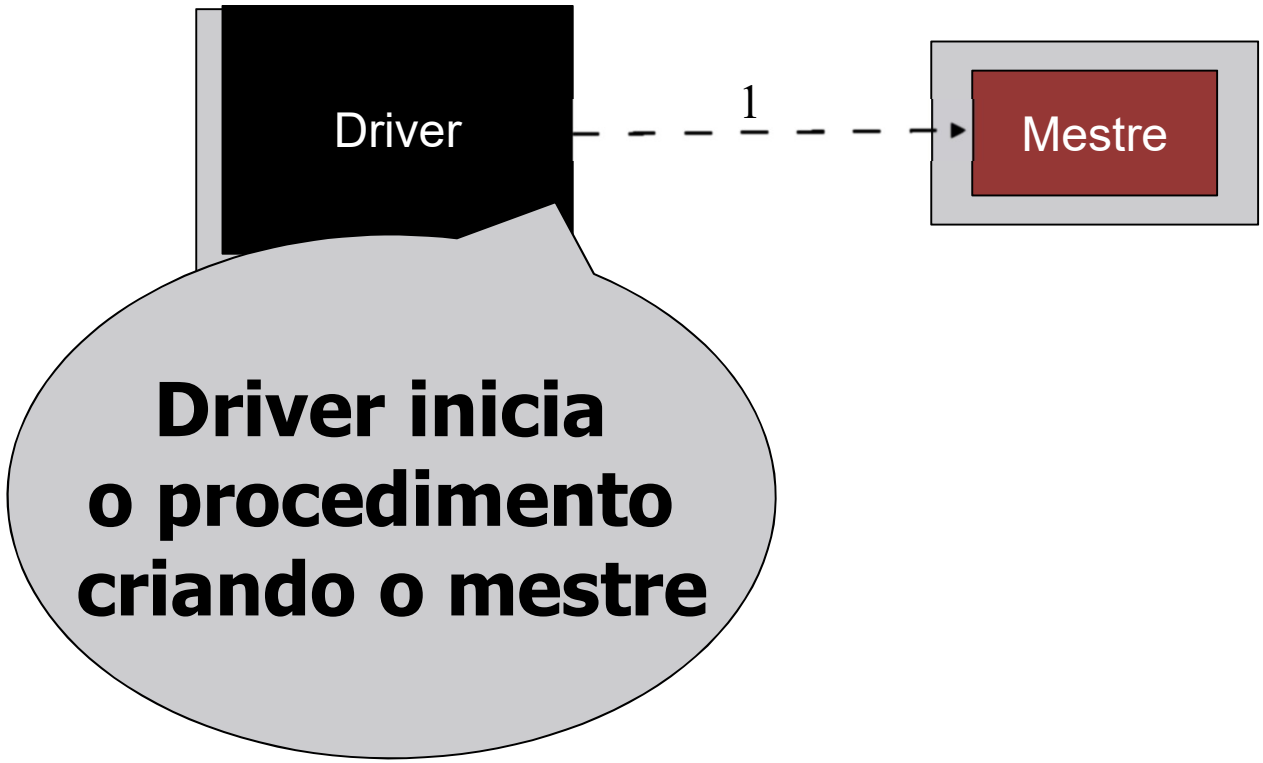
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Driver - programa** ou a linha de comando do Spark, inicia o procedimento se comunicando com o mestre
- **Mestre -** processo que recebe as instruções do driver e **gerencia os nós trabalhadores**, checa periodicamente se os trabalhadores estão operantes
- **Trabalhador –** um processo que recebe instruções do mestre, inicia e **gerencia um executor**, checa periodicamente se o executor está operante
- **Executor -** uma Máquina Virtual Java (JVM) que **abriga as tarefas**
- **Tarefa -** uma *thread* dentro da JVM que **processa uma partição** localmente

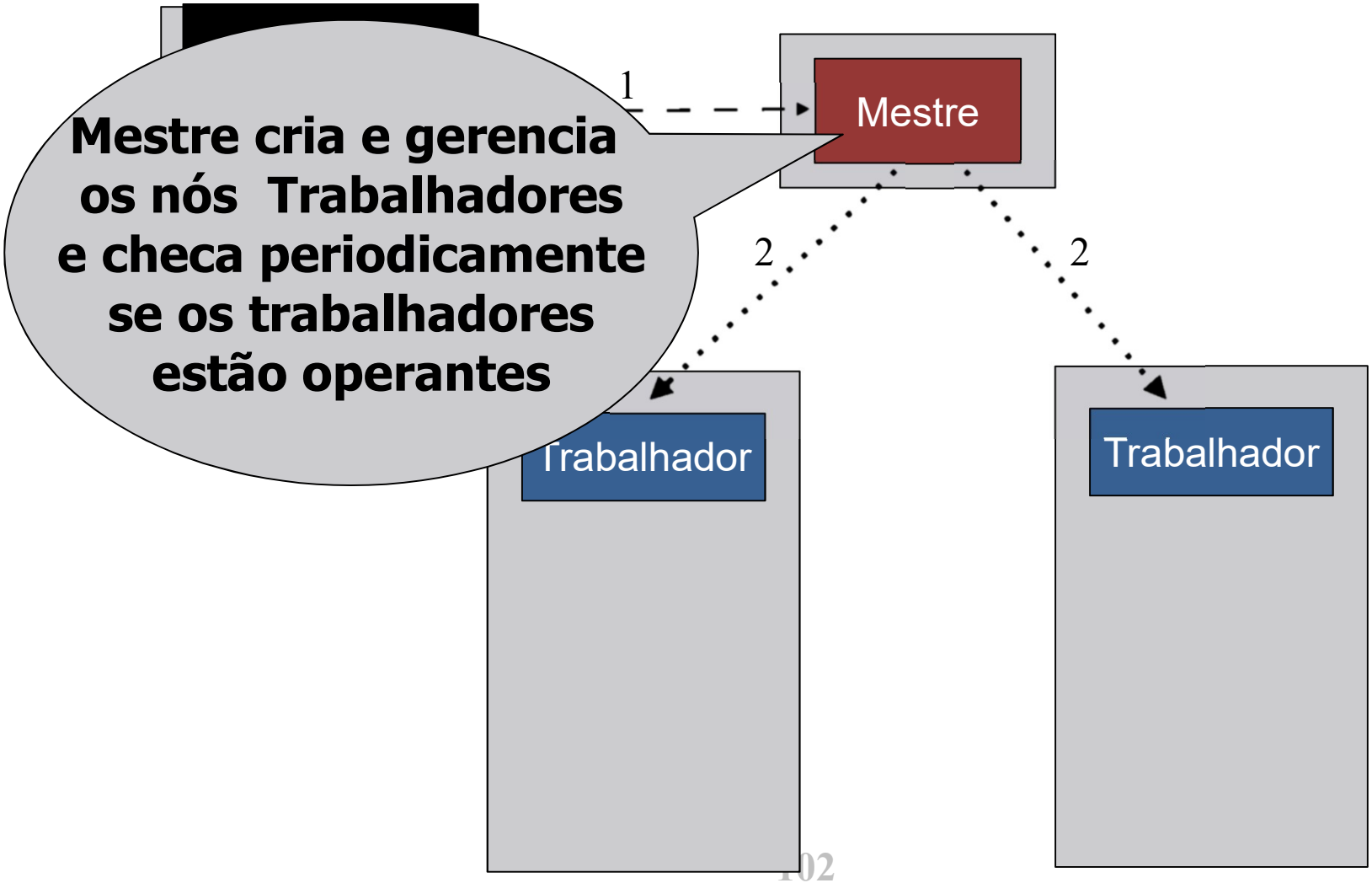
COPPE UFRJ

Apache Spark - Arquitetura

Apache Spark - Arquitetura



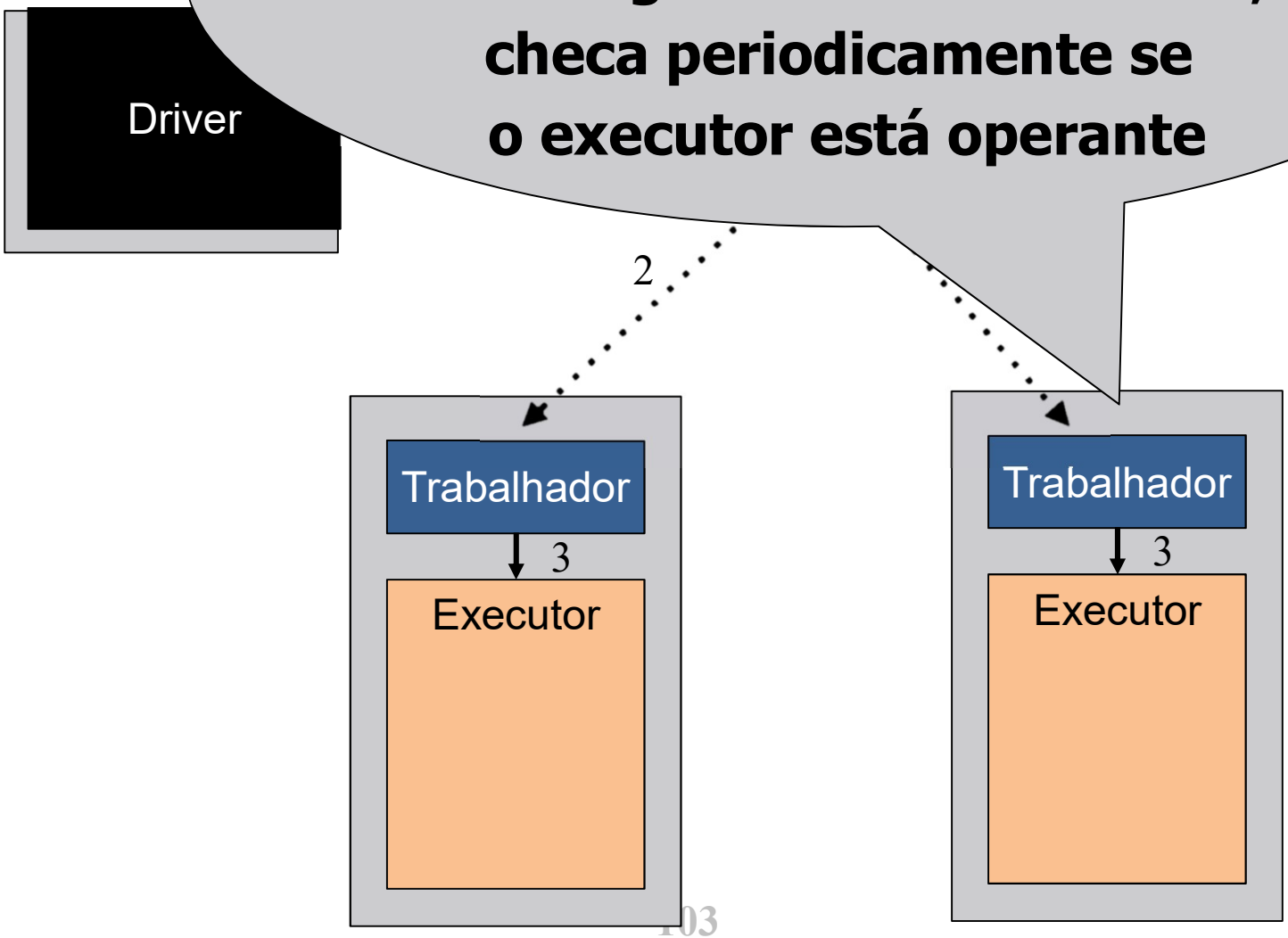
Apache Spark - Arquitetura



Apache Sp

Trabalhador

processo que recebe instruções do mestre, inicia e gerencia um executor, checa periodicamente se o executor está operante



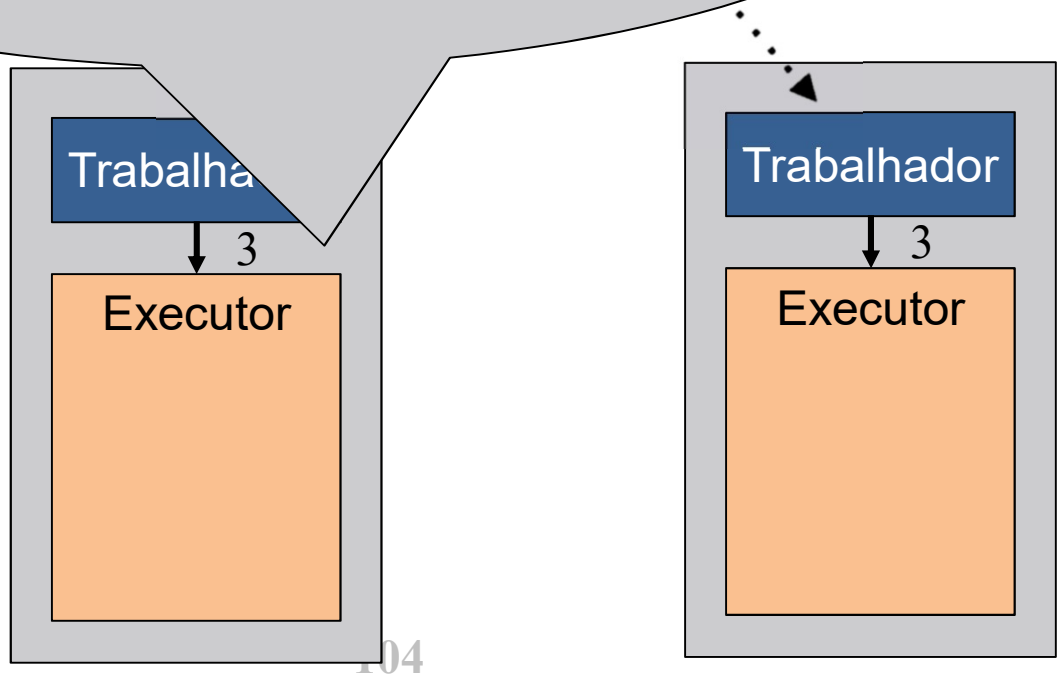
Apache Spark - Arquitetura

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Executor

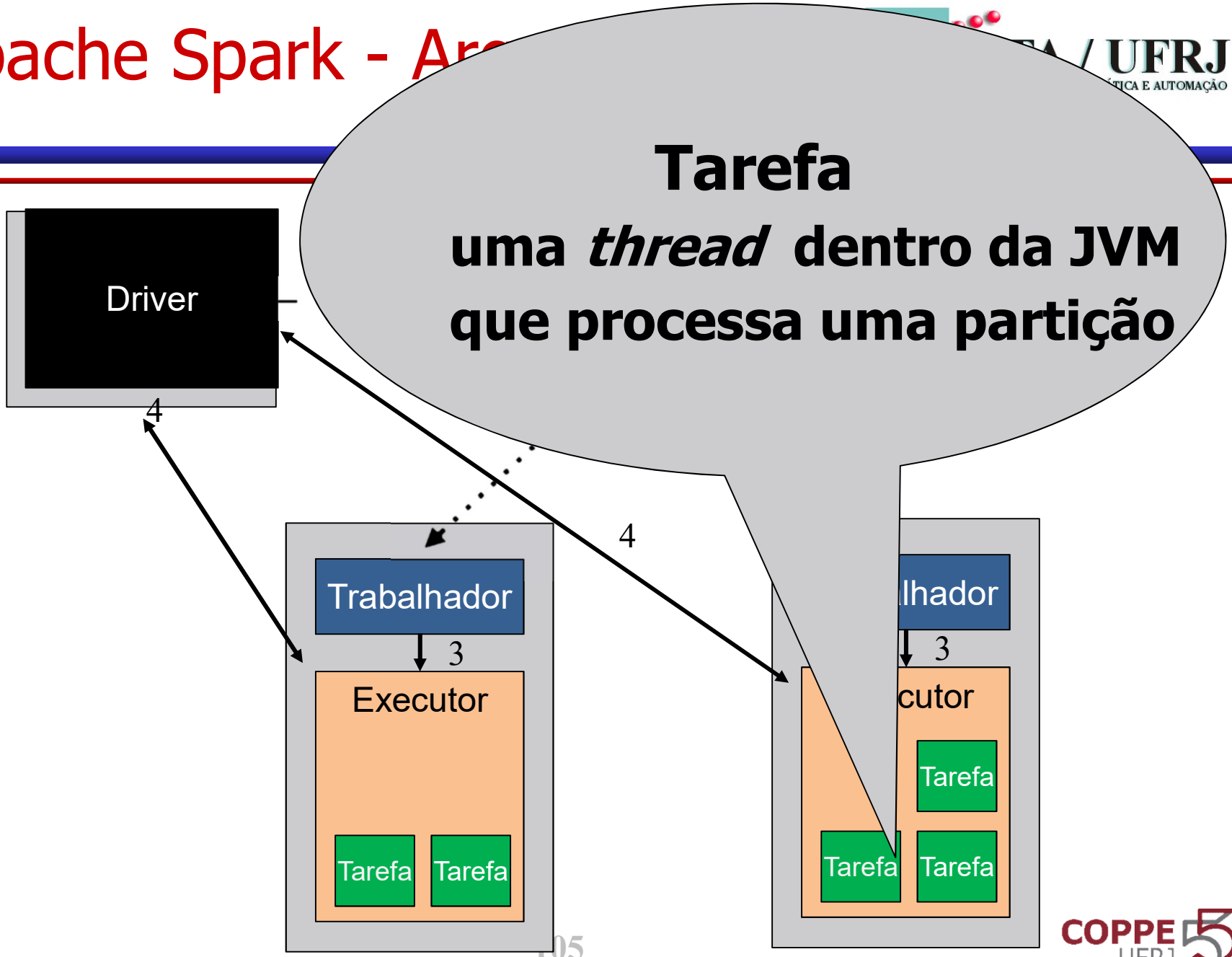
uma Máquina Virtual Java (JVM)

que realiza o processamento local



COPPE UFRJ

Apache Spark - Ar



Apache Spark – RDD distribuído - partições

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

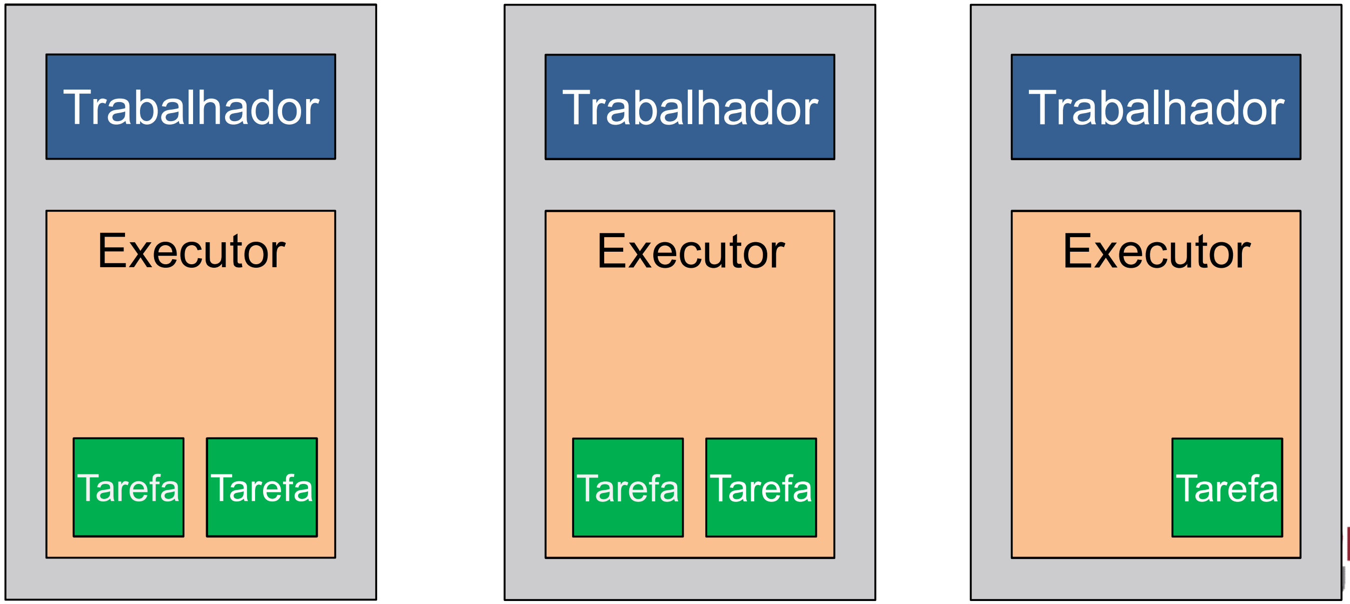
- **Resilient Distributed Datasets (RDD)**
  - Abstração de dados adaptada a ambientes de processamento distribuído
  - Distribuído
    - Divisão dos dados através de **partições**
    - Cada partição é enviada a uma tarefa
    - O usuário enxerga os dados como contínuos

COPPE UFRJ

Apache Spark – RDD distribuições de partições

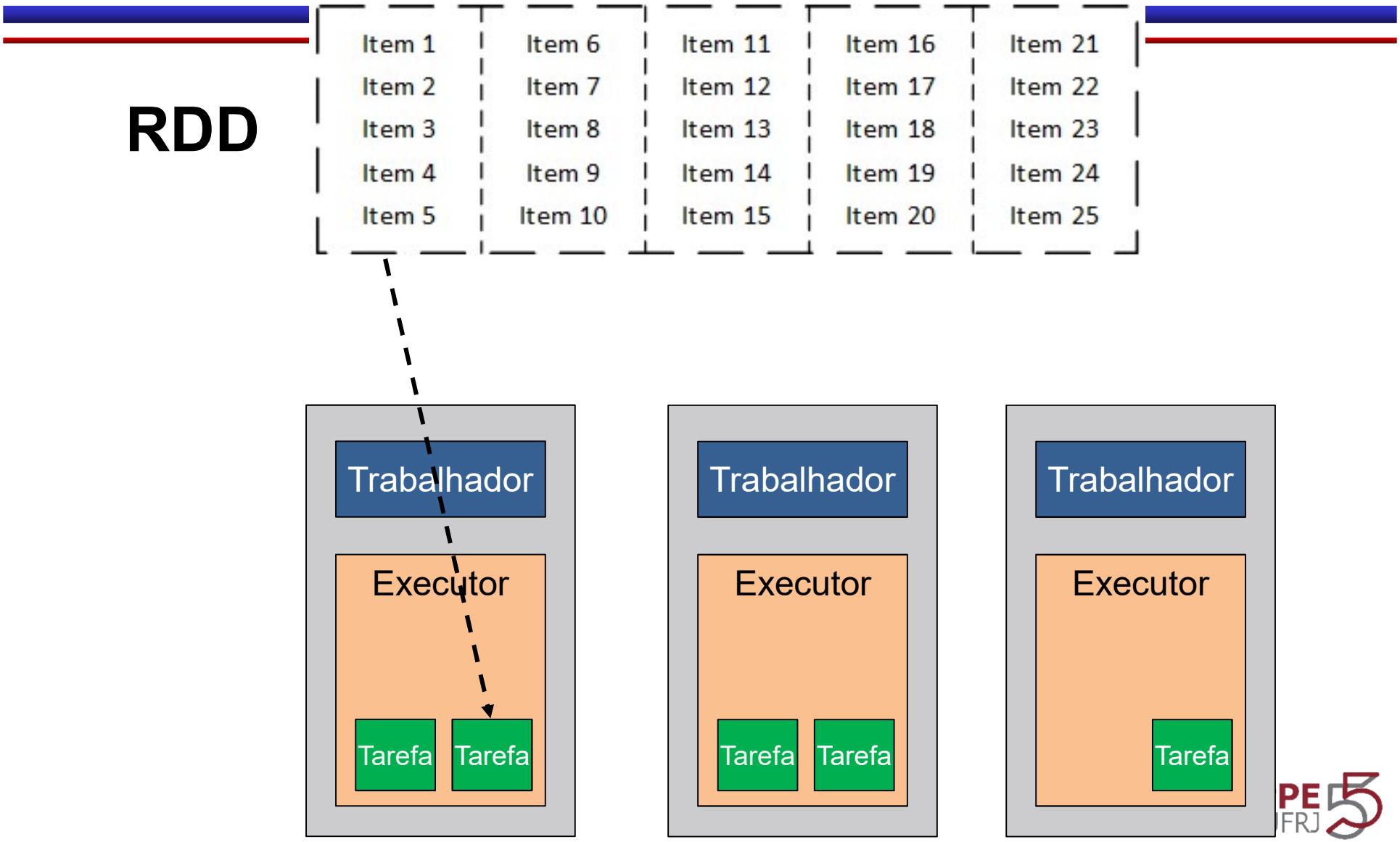
RDD

Item 1	Item 6	Item 11	Item 16	Item 21
Item 2	Item 7	Item 12	Item 17	Item 22
Item 3	Item 8	Item 13	Item 18	Item 23
Item 4	Item 9	Item 14	Item 19	Item 24
Item 5	Item 10	Item 15	Item 20	Item 25



Apache Spark – RDD distribuições de partições

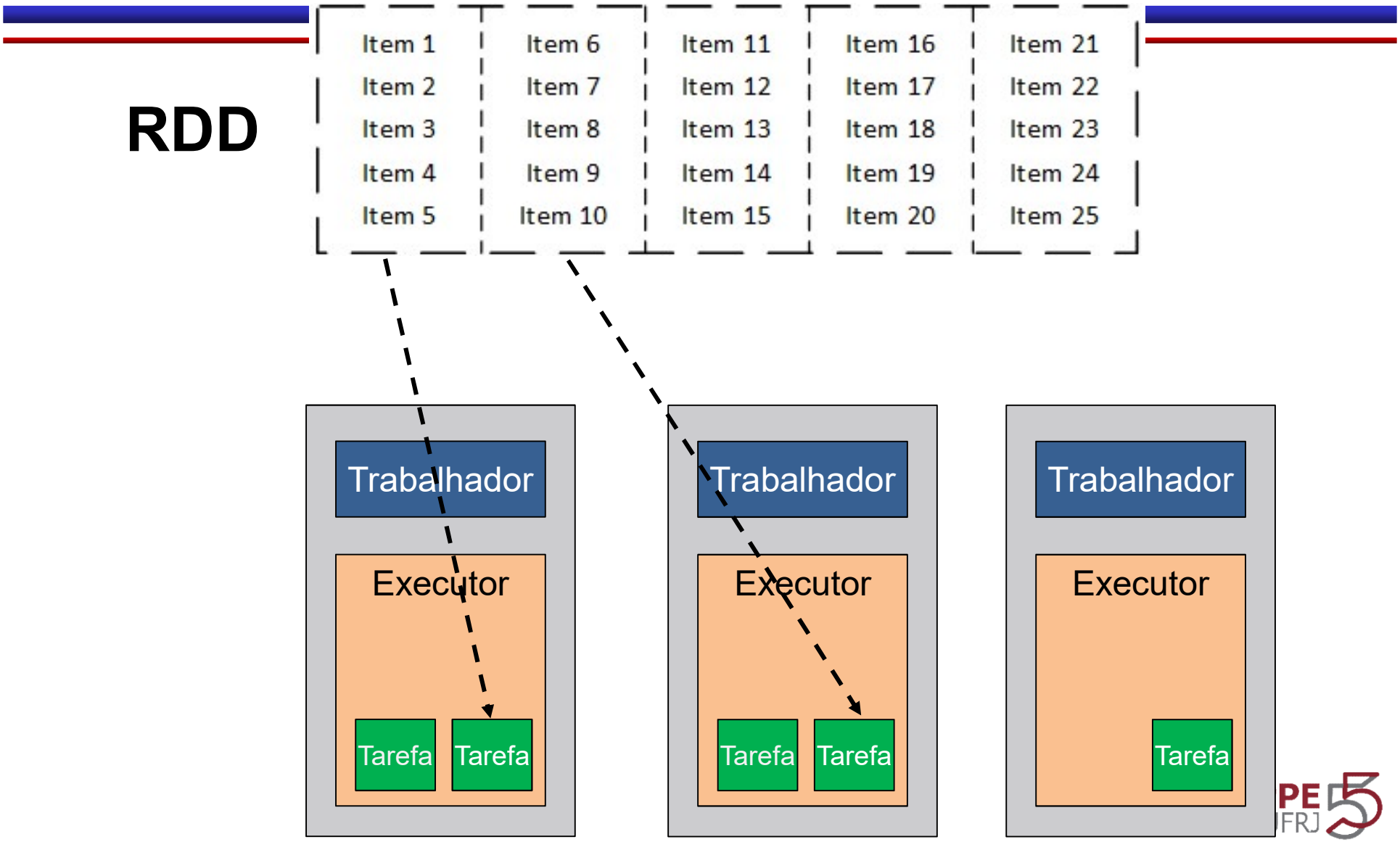
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



PE UFRJ

Apache Spark – RDD distribuições de partições

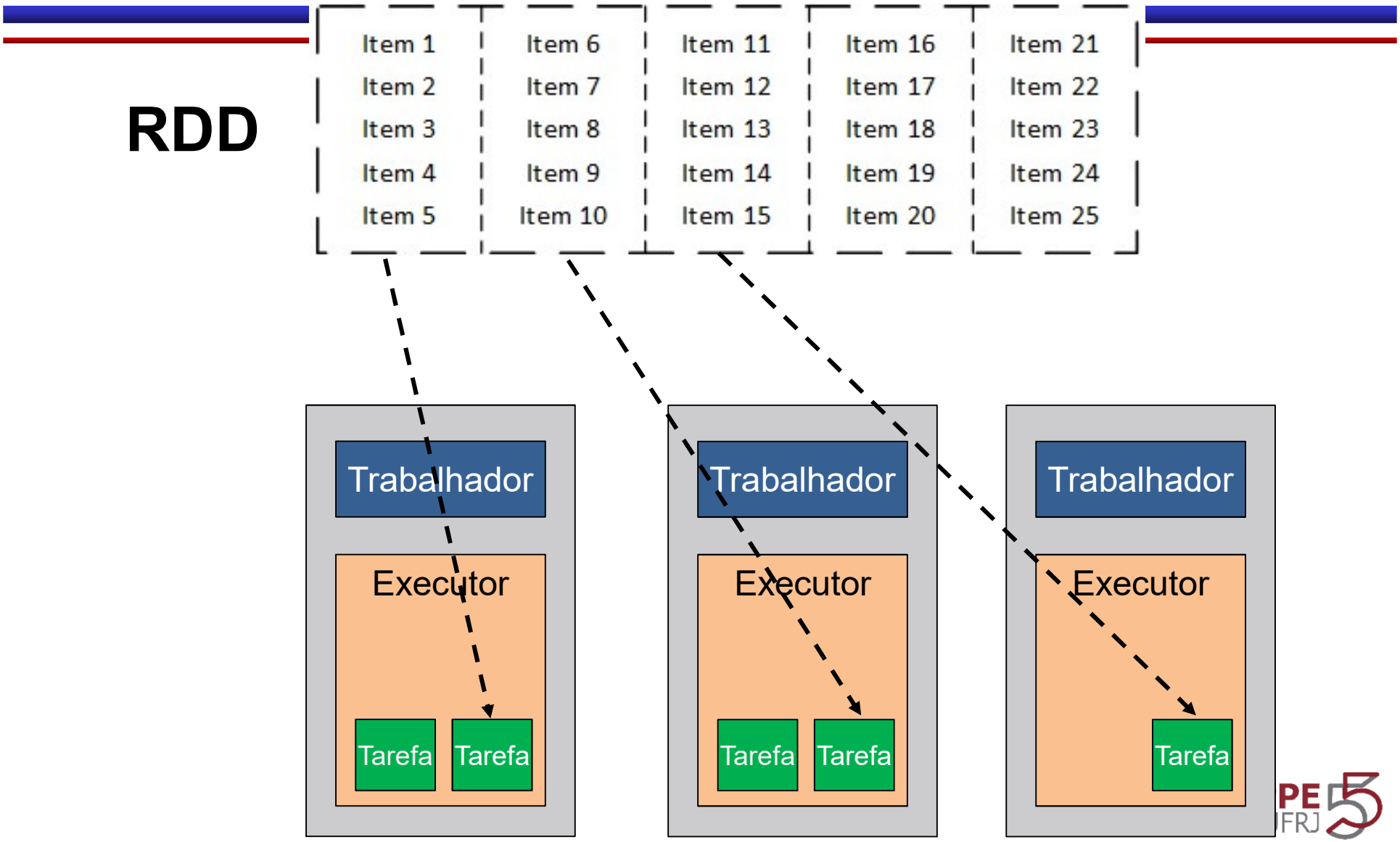
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



PE UFRJ

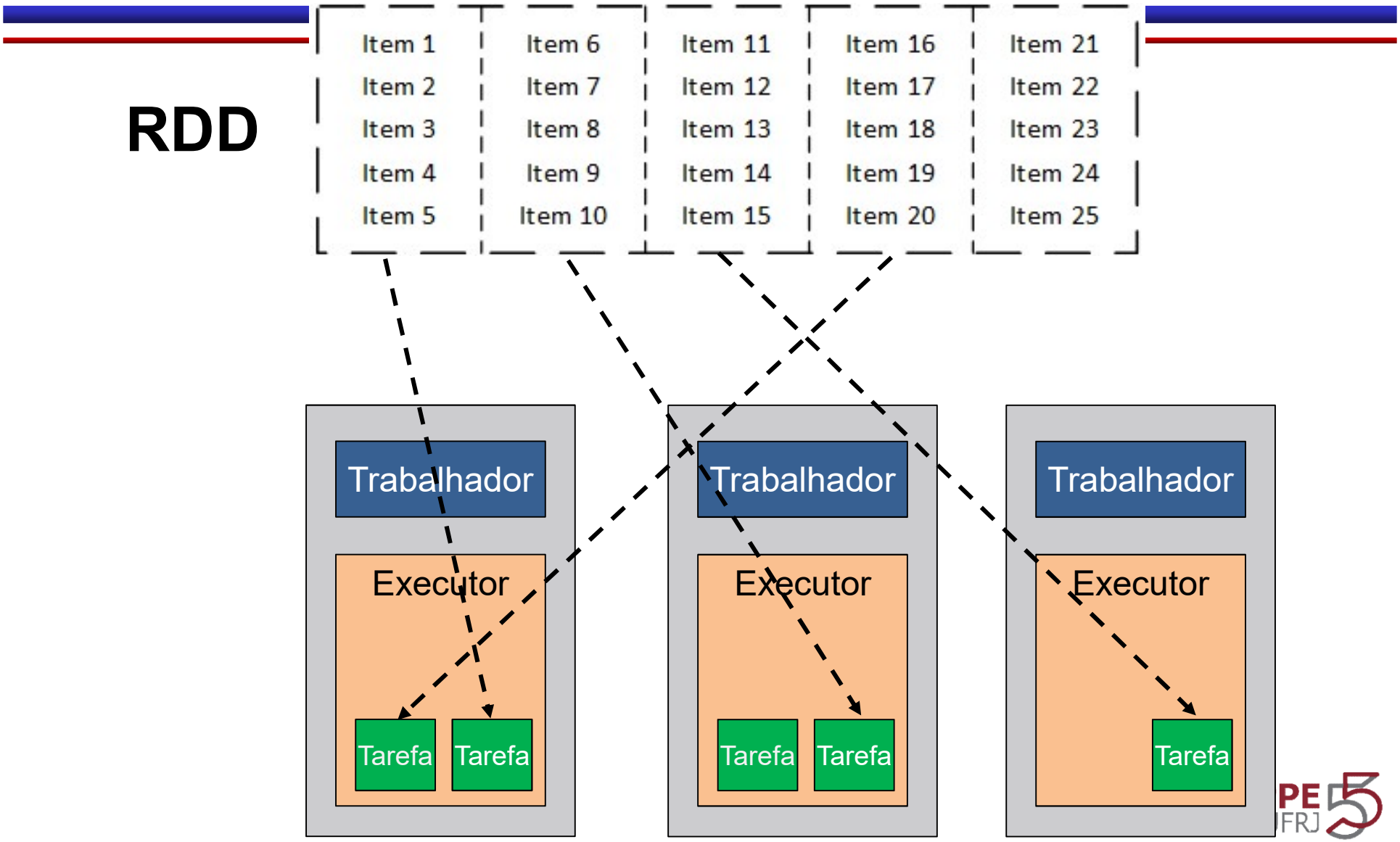
Apache Spark – RDD distribuições de partições

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Apache Spark – RDD distribuições de partições

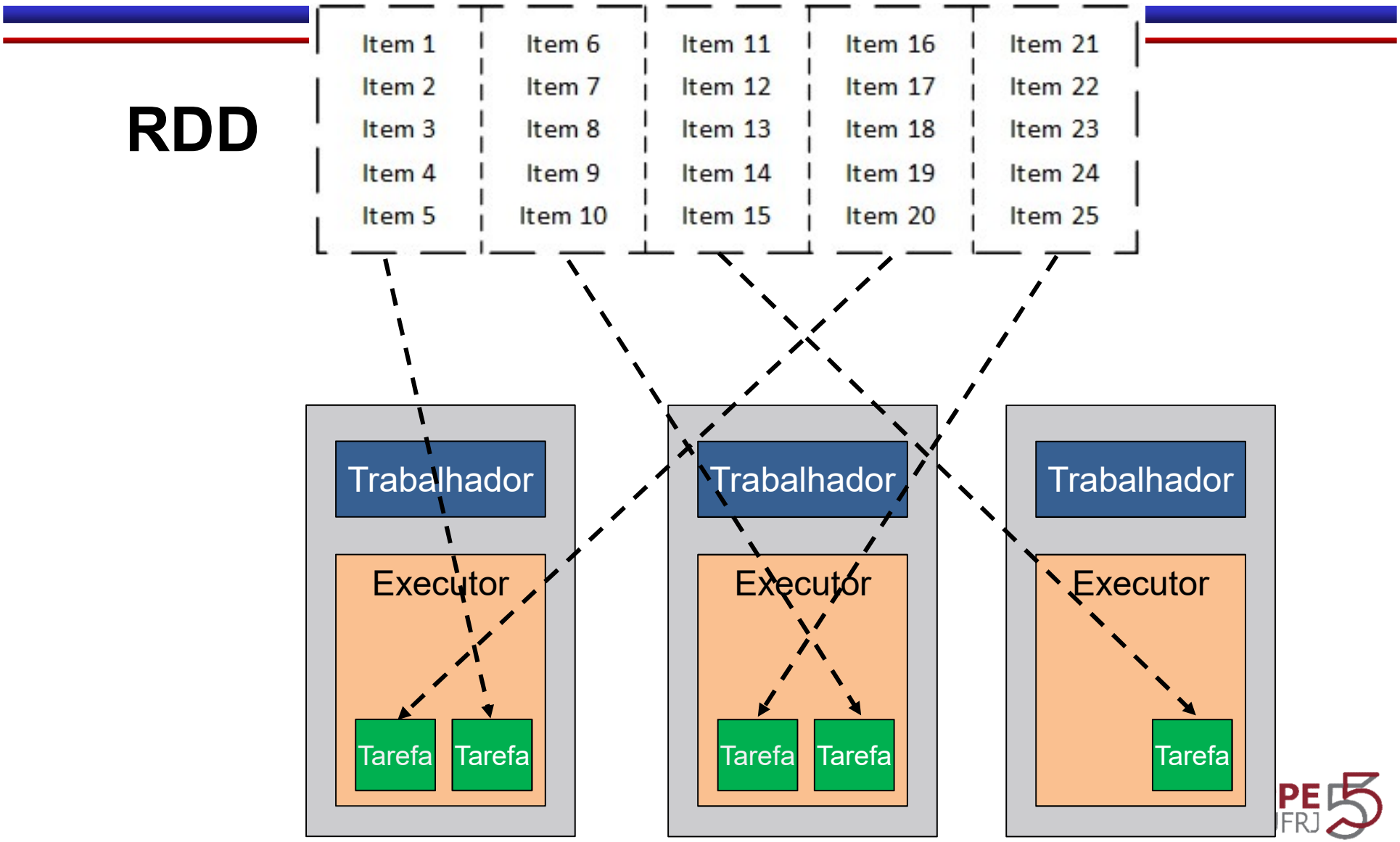
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



PE UFRJ

Apache Spark – RDD distribuições de partições

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

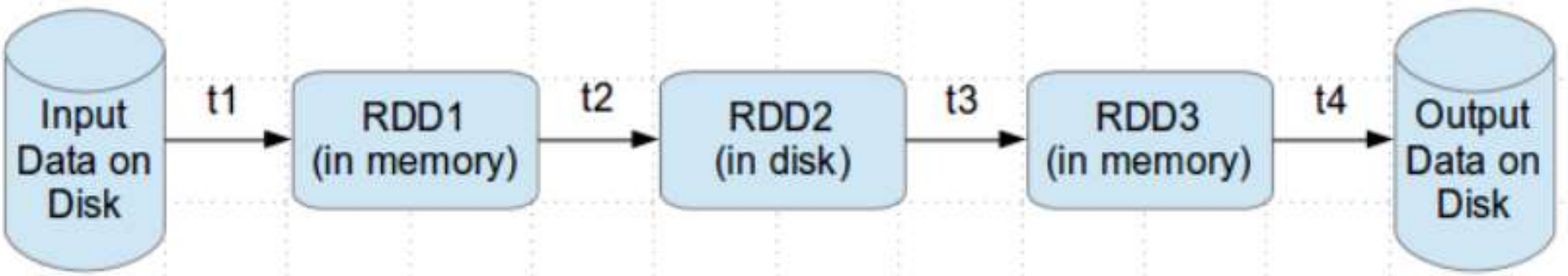


PE UFRJ

Apache Spark – RDD resiliência-tolerância a falhas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Resilient Distributed Datasets (RDD)**
  - Resiliente
    - **Imutável:** aplicar uma transformação cria um novo RDD
    - Para recuperar um RDD falho, **recomputa-se** seu histórico de transformações até a falha
    - Replicação de dados **não é necessária**



COPPE UFRJ

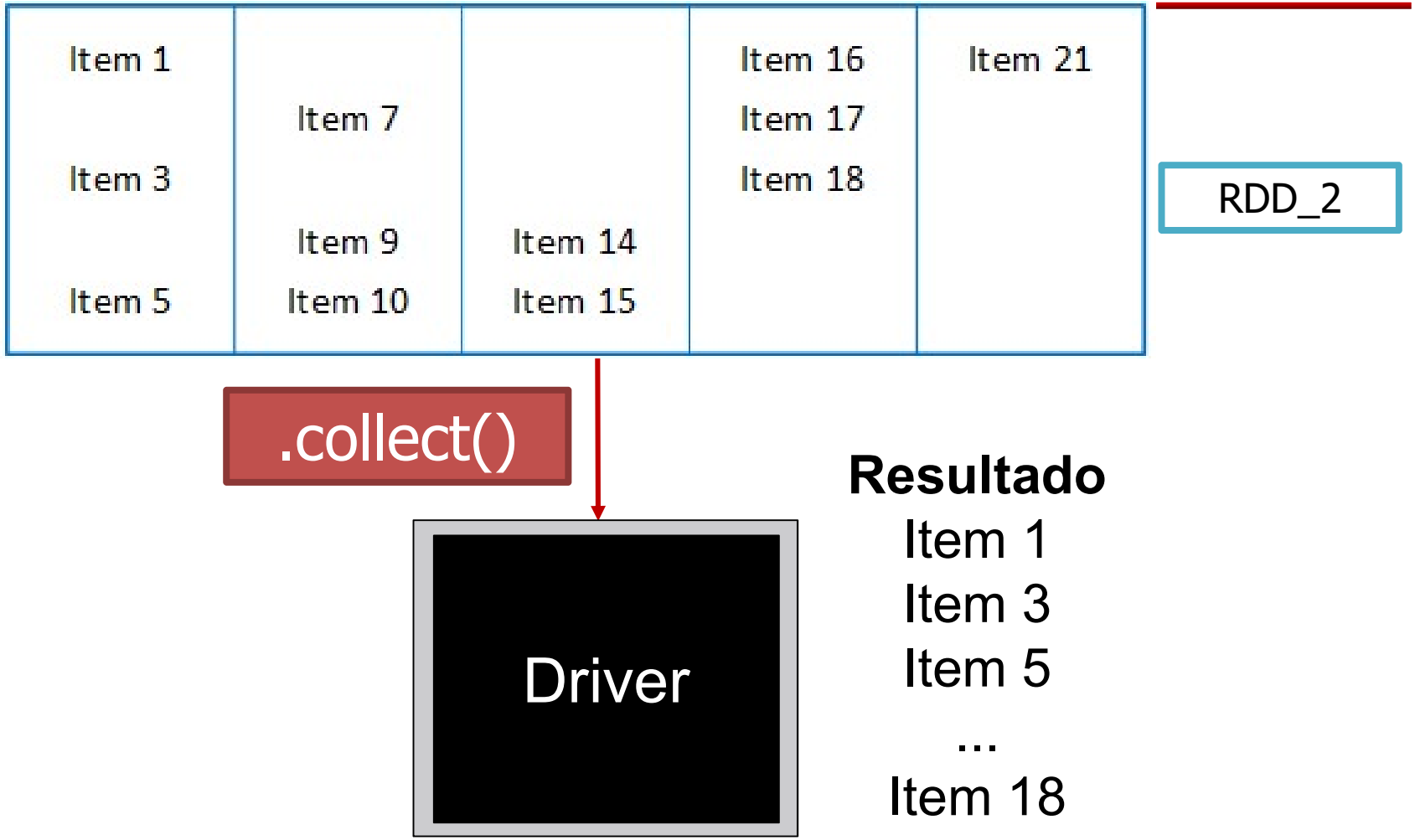
Apache Spark – RDD execução preguiçosa

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

COPPE UFRJ

Apache Spark – RDD coleta de resultado

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Apache Spark - DataFrames

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Dados estruturados**
- DataFrame: tabela ou matriz
  - Cabeçalho: nomes das características
  - Colunas: valores por característica
  - Linhas: amostras
- Construído sobre um RDD
  - Otimizações de bancos de dados relacionais
- Vantagem: maior **desempenho**
- Custo: estruturação dos dados

COPPE UFRJ

Apache Spark Streaming

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

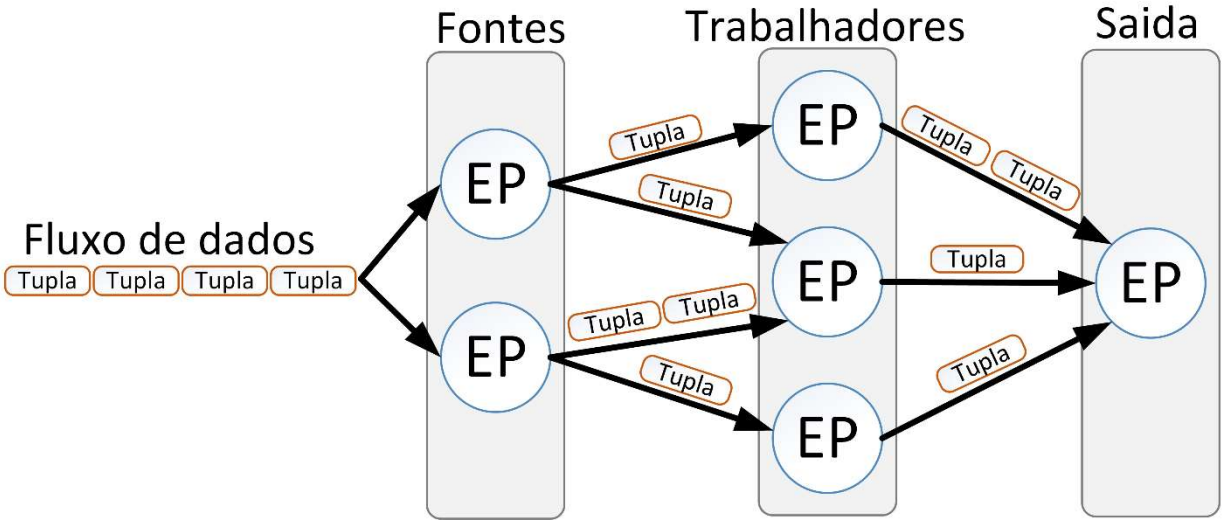
- **Hadoop e Spark**
  - Processamento em lotes
  - Conjunto de dados estático e conhecido
  - Inapropriado para fluxos e para tempo real
    - Resultados de segundos a horas
- **Spark Streaming**
  - Processamento em microlotes
  - Conjunto de dados dinâmico
  - Módulo p/ análise de fluxos em tempo quase real
    - Resultados em frações de segundo

COPPE UFRJ

Modelo de Processamento por Fluxo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Fonte de dados → Fluxo de dados ilimitado
  - Dados no formato de tuplas com marcação de tempo
  - Filas e *buffer* de dados
- Elementos de processamento
  - Topologia → Grafo direcionado acíclico (DAG)
    - Explícita vs. Implícita



COPPE UFRJ

Tempo para Processamento por Fluxo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Tempo do Evento
  - Tempo de geração dos dados na fonte
- Tempo de consumo de dados
  - Marcação de tempo em que o dado chega ao sistema de processamento
- Tempo de Processamento
  - Tempo em que o dado é de fato processado

Fonte: Tyler Akidau, disponível em https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101

COPPE UFRJ

Tempo para Processamento por Fluxo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Tempo do Evento
  - Tempo de geração dos dados na fonte
  - Tempo em que o dado é de fato processado

- Distorção: Diferença entre Tempos
- Fluxos desordenados

Fonte: Tyler Akidau, disponível em https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101

COPPE UFRJ

Apache Spark Streaming - DStreams

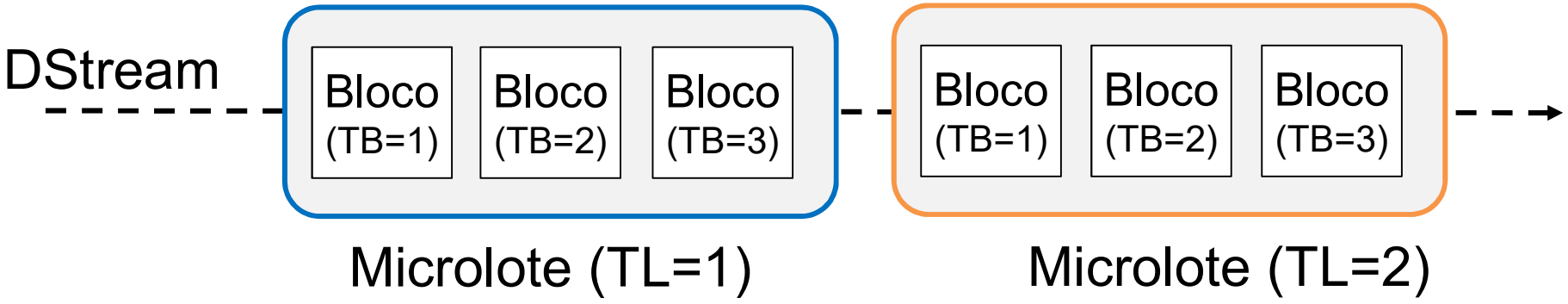
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Discretized Streams (DStreams)**
  - Transformar dados online em um micro *dataset* no formato que o Spark entende
  - Fluxo: sequência de **microlotes**
  - Cada microlote é uma janela de dados
  - Cada microlote possui **blocos**
    - Um bloco é uma subjanela de um microlote
    - **Objetivo:** distribuir os blocos entre tarefas para paralelizar o processamento de cada microlote

Apache Spark Streaming - DStreams

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Discretized Streams (DStreams)**
  - Um **bloco** é criado a cada tempo de fechamento de bloco
  - Um **microlote** é criado a cada tempo de fechamento de lote



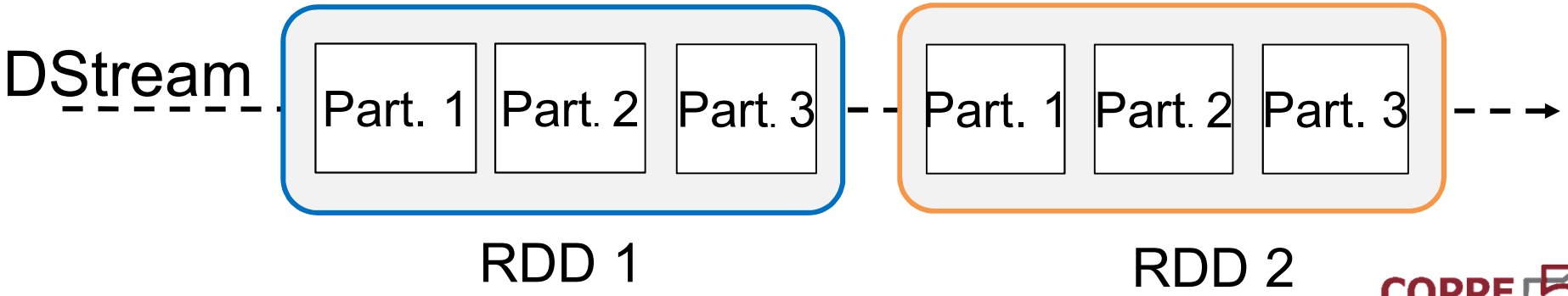
TL = tempo de fechamento de lote
TB = tempo de fechamento de bloco

COPPE UFRJ

Apache Spark - DStreams

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Discretized Streams (DStreams)**
  - Transformar dados online em um micro *dataset* no formato que o Spark entende
  - Microlotes→RDDs
  - Blocos →partições
  - DStream → sequência de RDD processada pelo Spark

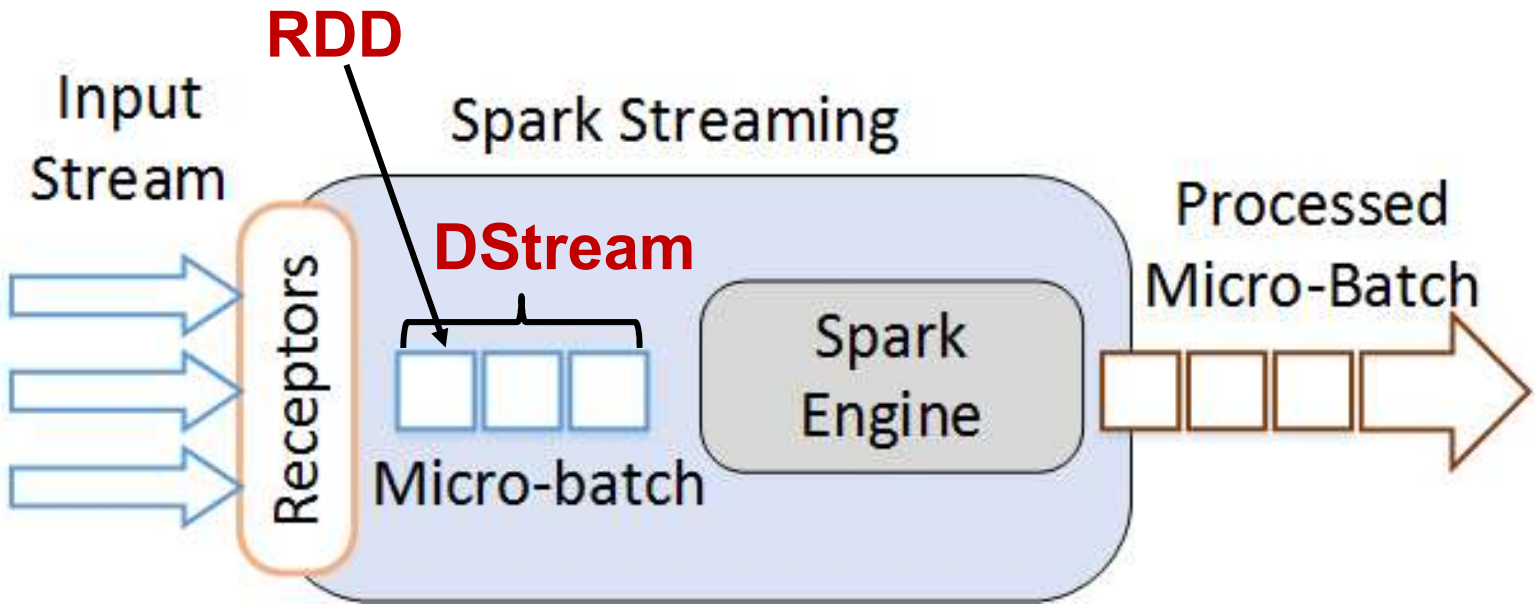


COPPE UFRJ

Apache Spark - Streaming

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Processamento em microlotes**



COPPE UFRJ

APACHE STORM

Apache Storm

APACHE STORM™

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Plataforma de processamento distribuído de fluxo nativa

- Criada por Nathan Marz, formado pela Universidade de Stanford e da equipe da BackType
- Empresa BackType comprada pelo Twitter (Jul/11)
- Storm disponibilizado Open Source (Set/2011)
- Incubada pelo Apache em setembro (2013)
- Projeto de Alto Nível da Apache em (2014)

COPPE UFRJ

Empresas Usando Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Entre outras...

COPPE UFRJ

Apache Storm

APACHE STORM™

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

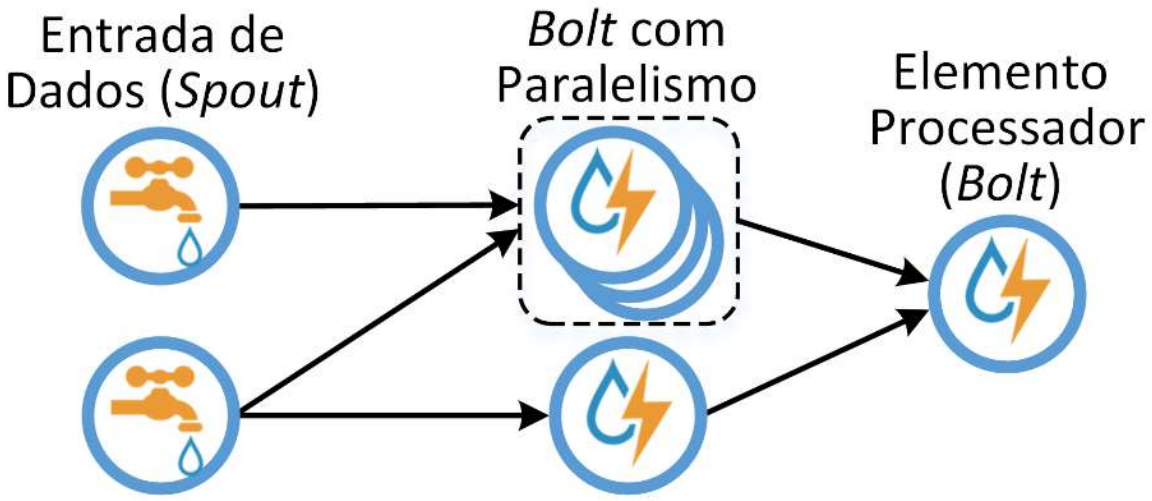
- Linguagem de programação Clojure
- Topologia de um gráfico acíclico direcionado
- Plataforma de Processamento Escalável de Fluxo
  - Computação baseada em Tuplas
    - Tuplas com ID (*Named tuple)*
  - **Fluxo** → Sequência ilimitada de tuplas (itens de dados)
  - Programas representados em **topologias** → Grafo Acíclico Direcionado de operadores sobre o fluxo
    - **Vértices** → computação e transformação de dados
    - **Arestas** → Fluxo de dados entre nós de computação

COPPE UFRJ

Topologia
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Topologia de processamento distribuído explícita
  - Processamento por fluxo nativo (verdadeiro)
    - Tratamento dos dados Tupla a Tupla
  - **Spout (bica)** → Entrada de dados
    - Inserção de novas tuplas no sistema
  - **Bolt (relâmpago)** → Nó de processamento

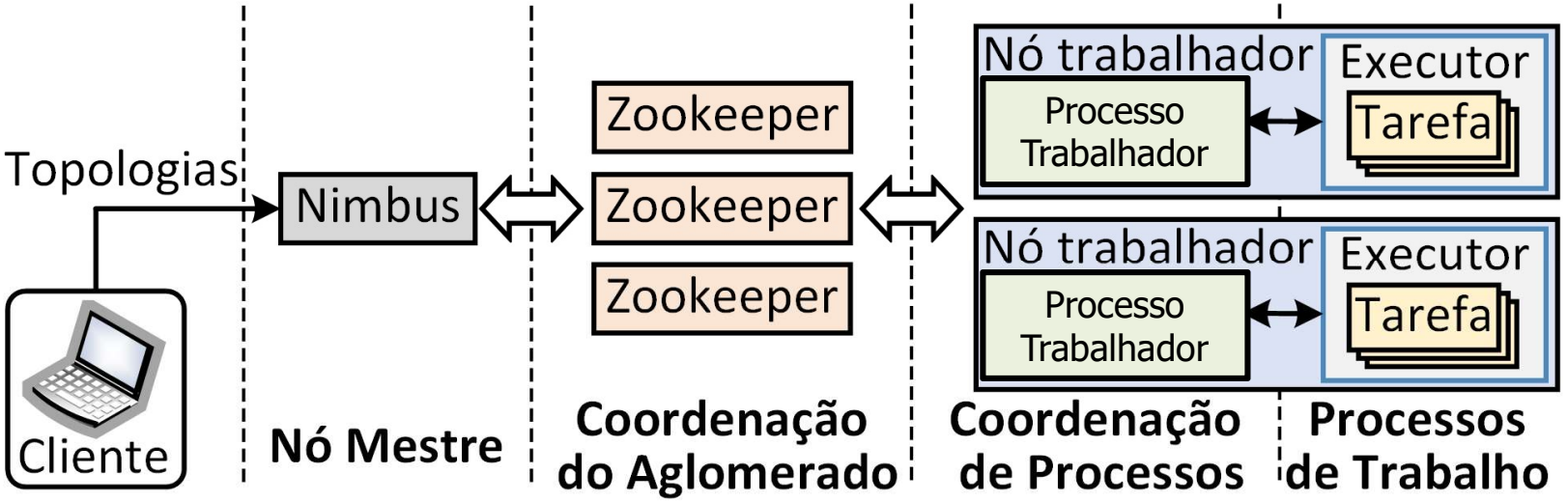


COPPE UFRJ

Arquitetura
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Dois tipos de nós no aglomerado Storm
  - **Nó Mestre (NIMBUS)**
    - Escalonamento, orquestração e monitoramento de falhas
  - **Nó Trabalhador (Supervisor)**
    - Execução do processo Trabalhador → computação da aplicação
- **ZooKeeper** → Coordenação entre **NIMBUS** e **Supervisores**
  - Alta disponibilidade, dados compartilhados e técnicas de sincronização



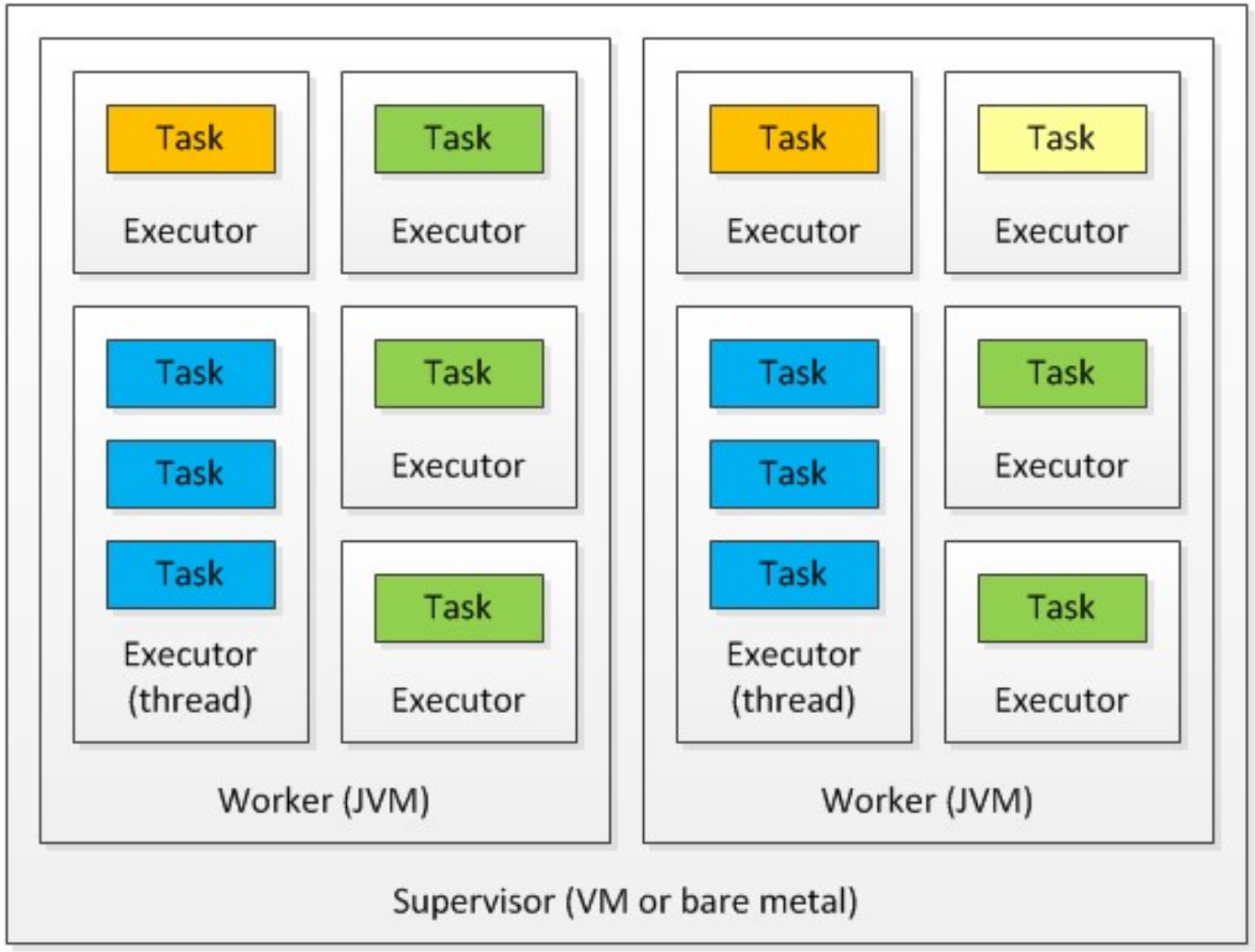
Arquitetura
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Processo Trabalhador** → provê paralelismo entre topologias
  - Pelo menos um por aglomerado e cada um pertence a somente uma topologia
  - Tolerância a falhas e isolamento
- **Executor** → provê paralelismo interno a uma topologia
  - *Thread* no processo Trabalhador
  - Execução de uma ou mais tarefas do mesmo componente (spout ou bolt)
- **Tarefa** → realiza o processamento dos dados
  - Réplica de um componente (spout ou bolt)

Arquitetura
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

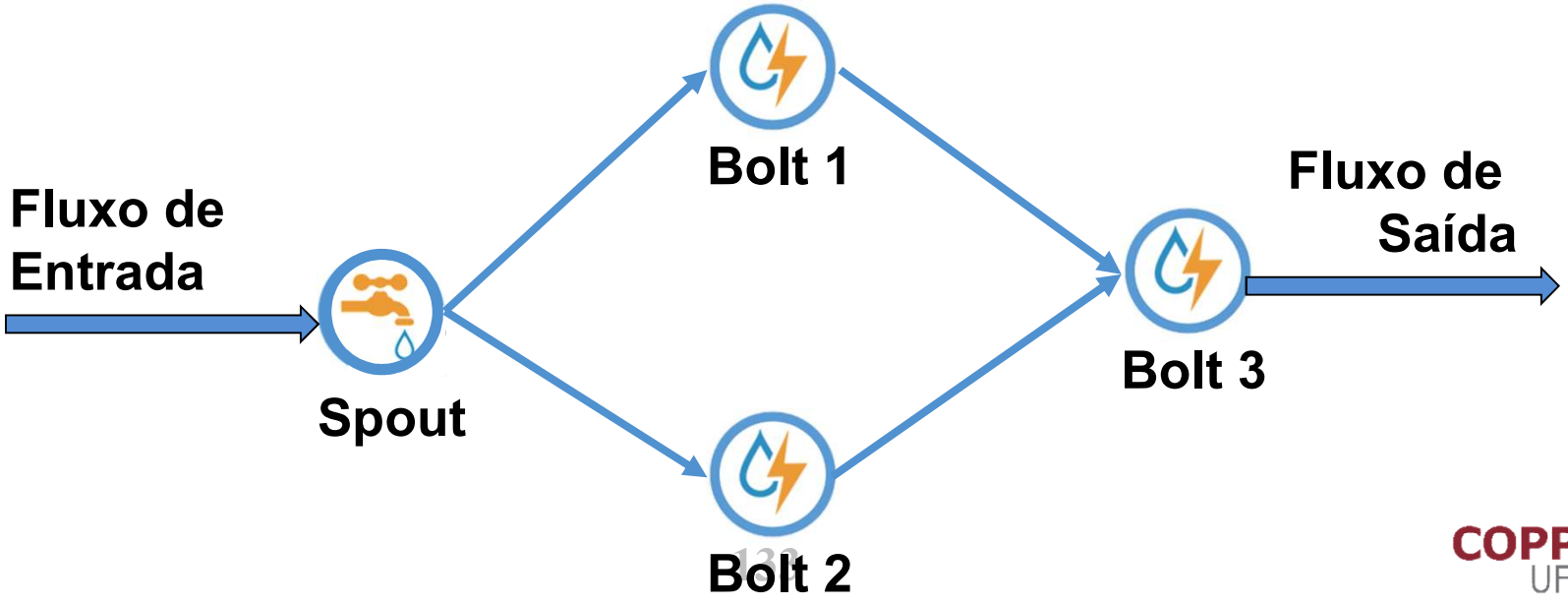


COPPE UFRJ

Exemplo Simples Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Cálculo da média aritmética do fluxo de entrada
  - Topologia explícita simples → processamento em paralelo
    - Um *spout*
    - Dois *bolts* em paralelo
    - Um *bolt* de convergência

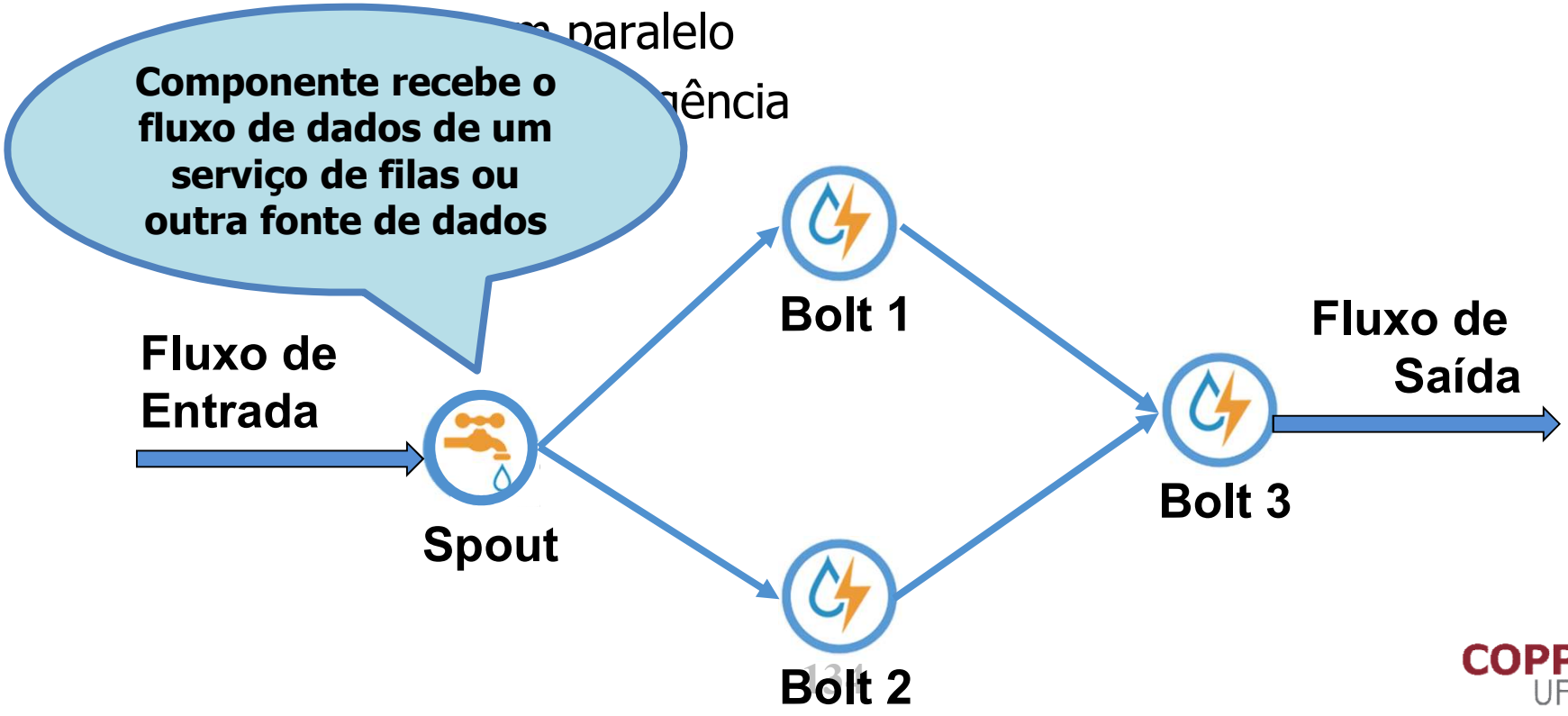


COPPE UFRJ

Exemplo Simples Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Cálculo da média aritmética do fluxo de entrada
  - Topologia explícita simples → processamento em paralelo
    - Um *spout*
    - paralelo
    - ência

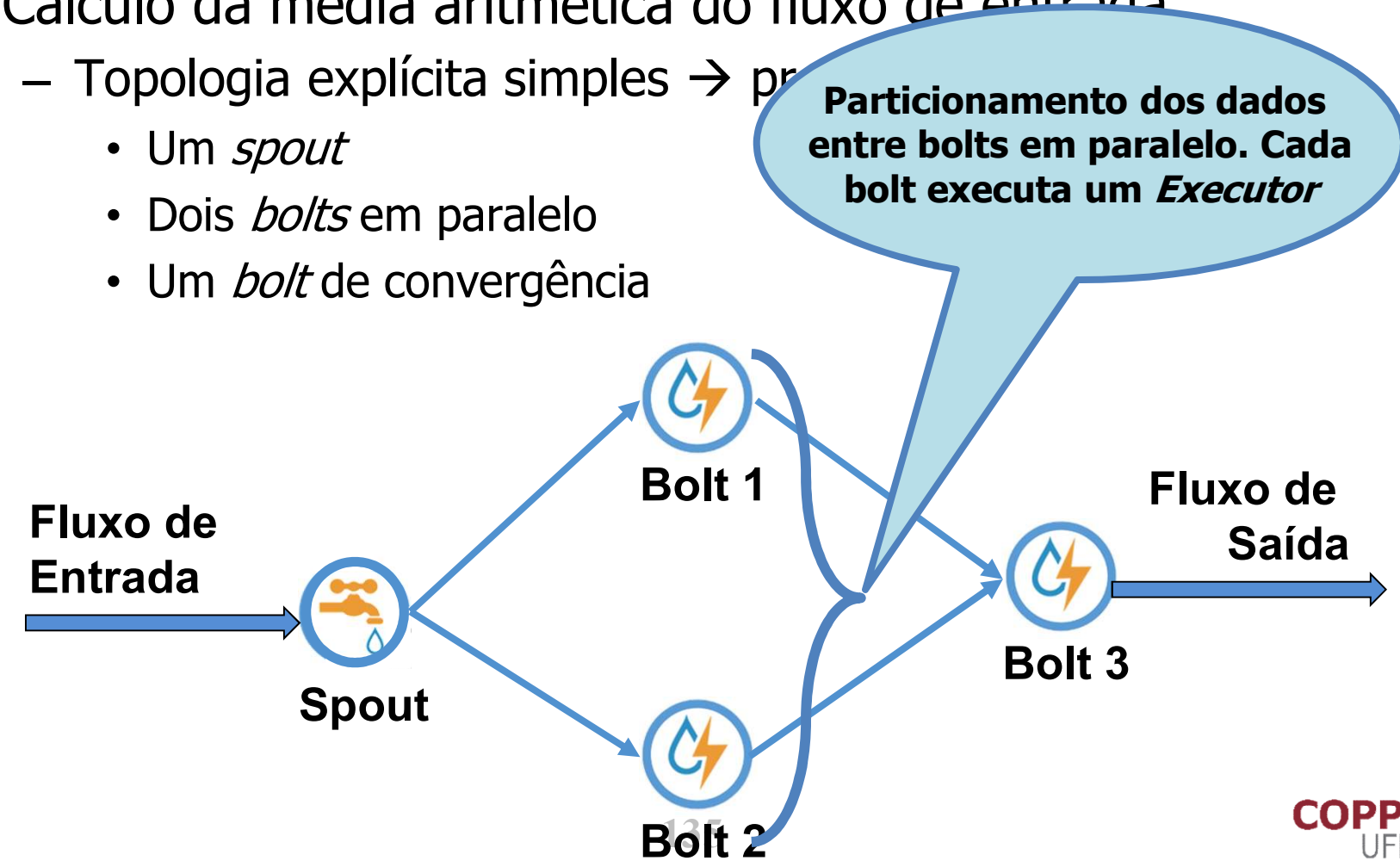


COPPE UFRJ

Exemplo Simples Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Cálculo da média aritmética do fluxo de entrada
  - Topologia explícita simples → pr
    - Um *spout*
    - Dois *bolts* em paralelo
    - Um *bolt* de convergência



COPPE UFRJ

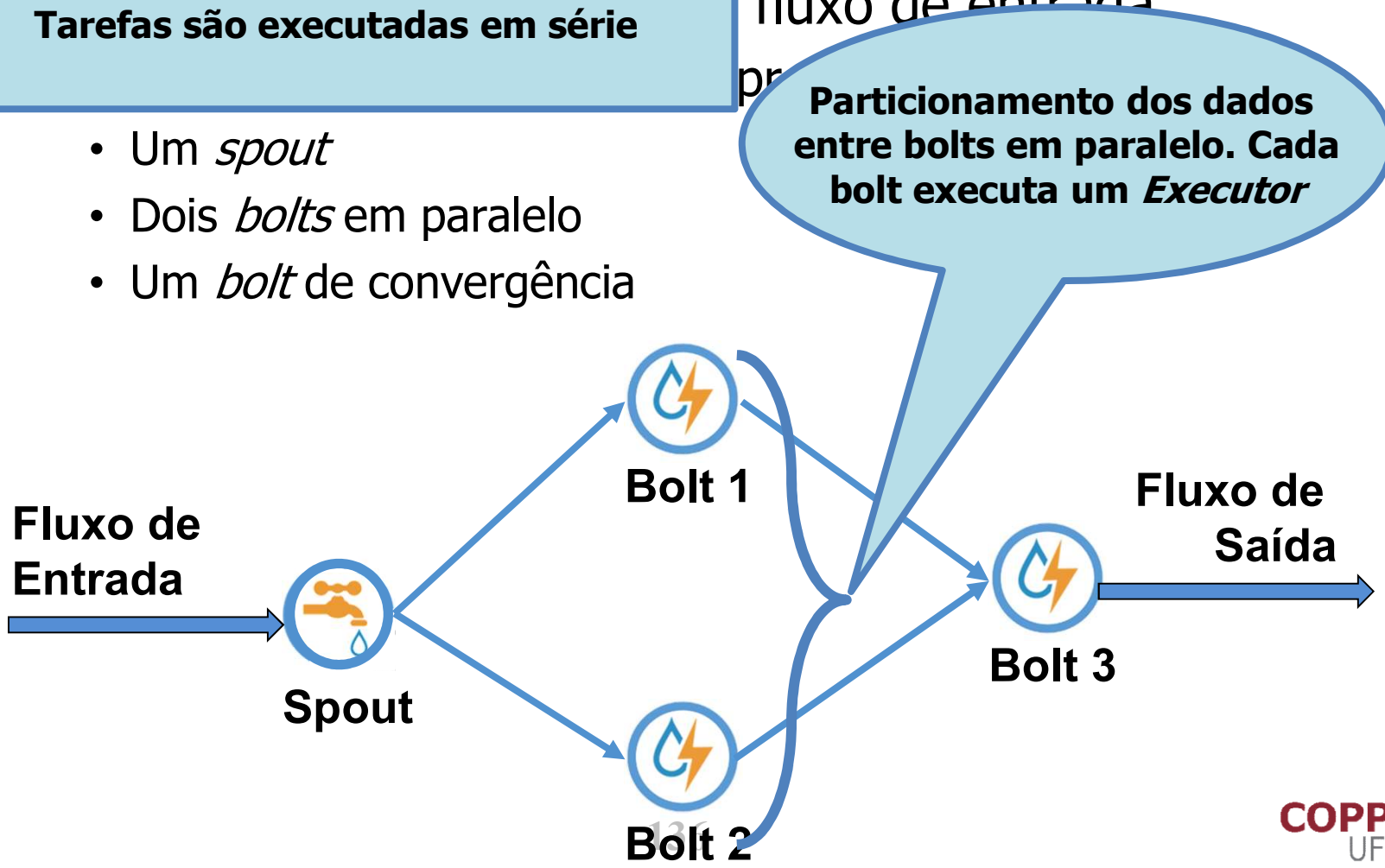
Paralelismo pode ocorrer:

- **Entre processos → Cada *bolt* executa em um processo trabalhador diferente**
- **Entre threads → Cada *bolt* executa em uma thread *Executor* dentro do processo trabalhador**
  - **Tarefas são executadas em série**

fluxo de entrada

pr

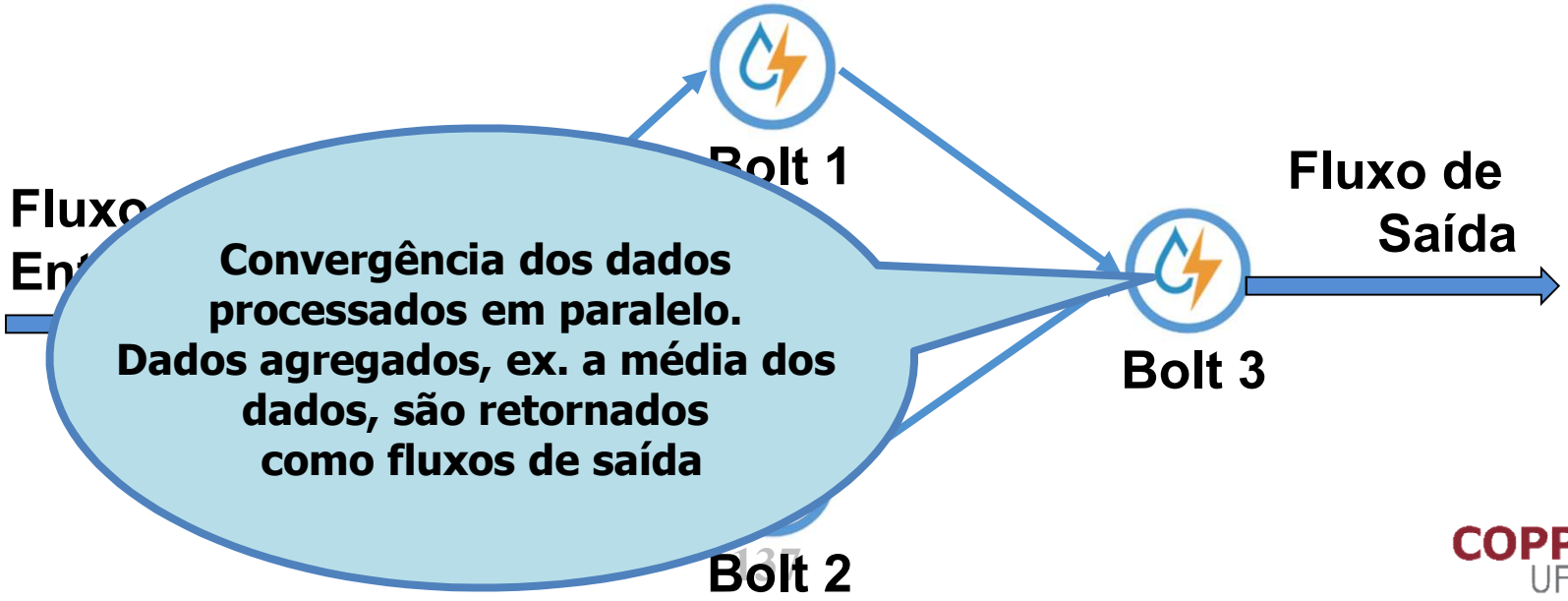
- Um *spout*
- Dois *bolts* em paralelo
- Um *bolt* de convergência



Exemplo Simples Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Cálculo da média aritmética do fluxo de entrada
  - Topologia explícita simples → processamento em paralelo
    - Um *spout*
    - Dois *bolts* em paralelo
    - Um *bolt* de convergência

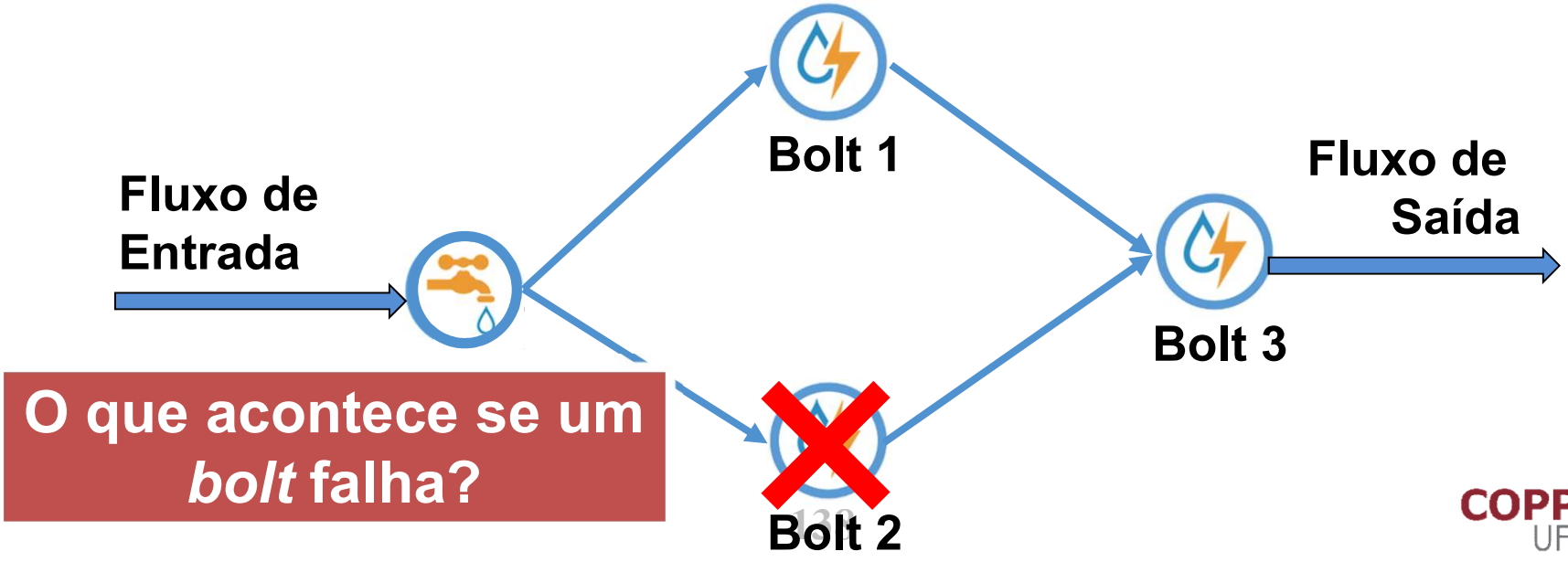


COPPE UFRJ

Exemplo Simples Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

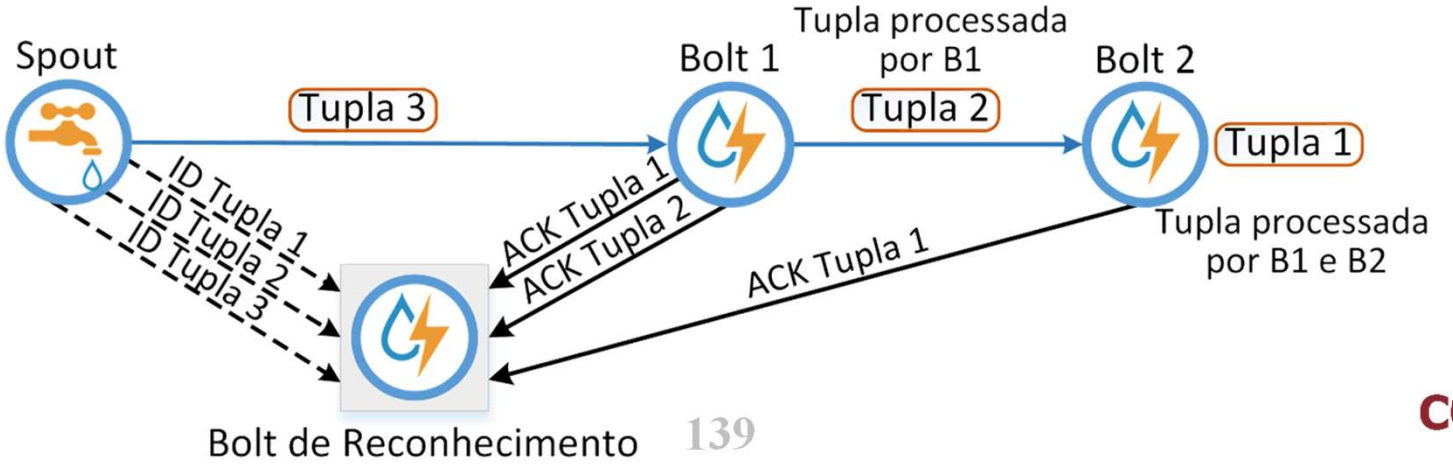
- Cálculo da média aritmética do fluxo de entrada
  - Topologia explícita simples → processamento em paralelo
    - Um *spout*
    - Dois *bolts* em paralelo
    - Um *bolt* de convergência



Tolerância a Falhas
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Semântica de Tolerância a Falhas
  - *"At least once"* → Cada tupla é processada pelo menos uma vez
    - Fluxos completamente processados
  - *Acker Bolt* → Registra o progresso de cada tupla na topologia Storm



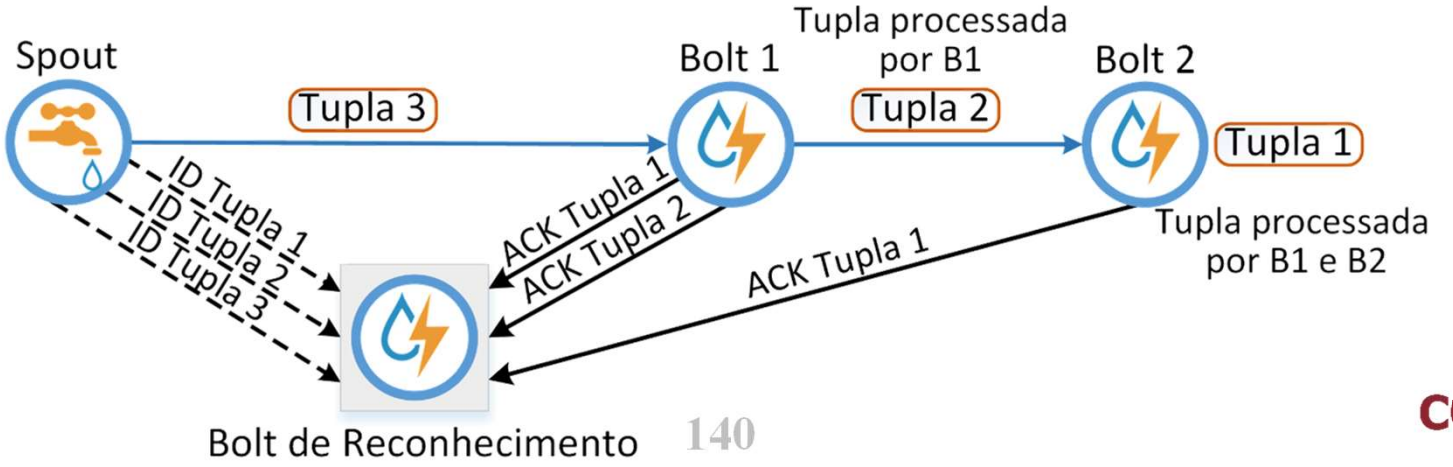
COPPE UFRJ

Tolerância a Falhas
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Semântica de Tolerância a Falhas
  - *"At least once"* → Cada tupla é processada pelo menos uma vez
    - Fluxos completamente processados
  - *Acker Bolt* → Registra o progresso de cada tupla na topologia Storm

Duplicação de Tuplas → Falha em um bolt em paralelo faz com que tuplas de todos os demais bolts em paralelo sejam reprocessadas



COPPE UFRJ

Tolerância a Falhas
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Possíveis Falhas
  - Tupla não reconhecida por que a tarefa falhou
    - A identificação da Tupla expira na raiz da árvore e, então, é reemitida
  - Bolt de ACK (Acker) falha
    - Todas as tuplas que o acker estava mantendo o estado expirarm e têm que ser reemitidas
  - Spout Falha
    - A fonte que é representada pelo spout é responsável por reemitir os dados.
      - Ex: Kestrel, RabbitMQ, Kafka
- Possibilidade de duplicação de tuplas processadas
  - Tupla processada e não reconhecida é reprocessada

COPPE UFRJ

Trident
Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Abstração de alto nível para o Storm
  - Criação de topologias implícitas
    - Mais complexas
  - Compilação de topologias Trident em topologias Storm eficientes
  - Micro-batch
  - Abstração em funções executadas sobre as tuplas → Similar ao Map/Reduce
    - Map, Filter, Windowing, Merges and Joins
- Gravação de Estados do Processamento
  - *Stateful*
  - Memória ou Disco
    - Memcached ou Cassandra DB → HDFS

COPPE UFRJ

Tolerância a Falhas
Trident - Apache Storm

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Semânticas de Tolerância a Falha
  - *At least once*
    - Garantia de processamento de todas as tuplas com consistência eventual
      - Nativo do Storm
  - ***Exactly once***
    - Garantia de consistência forte
      - Todas as tuplas são processadas somente uma vez
    - Estado de processamento em armazenamento persistente
      - HDFS

APACHE FLINK

Apache Flink - História

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **2008**
  - Ideia concebida pelo Professor Volker Markl – Technische Universität Berlin
- **2010**
  - Projeto alemão "Stratosphere: Information Management on the Cloud"
  - Colaboração da Technische Universität Berlin, Humboldt-Universität zu Berlin e Hasso-Plattner-Institut Potsdam
  - Financiado pela German Research Foundation (DFG)
- **Mar/2014**
  - Incubado pela Apache
- **Dez/2014**
  - Projeto de Alto Nível (TLP) da Apache

Apache Flink – Empresas usando Flink

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Fonte: https://cwiki.apache.org/confluence/display/FLINK/Powered+by+Flink

COPPE UFRJ

Apache Flink– A plataforma

- Plataforma código aberto para processamento distribuído
  - Mais de 380 contribuidores no Github
- Processamento híbrido
  - Processamento por fluxos (nativo)
  - Processamento em lotes (caso especial)
- Processamento com estados
  - Aplicações podem manter resumos do processamento no tempo

Apache Flink - A plataforma

- Implementada em Java
  - Mas provê API para Scala
- Diversas APIs
  - DataSet API – Proecessamento em lotes
  - DataStream API → Processamento por fluxo
  - Table API → Consultas relacionais
- Bibliotecas específicas
  - FlinkML → Aprendizado de máquina para Flink
  - Gelly → Biblioteca de grafos para Flink
- Terminal interativo para análise de dados

Apache Flink - Dados

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Tipo de dado – DataStream
  - Abstração do fluxo de dados
    - Análoga à Tupla do Storm
  - Sequência de registros parcialmente ordenados e infinitos
    - Ordem não é garantida caso operador receba dois DataStreams
  - Operações em DataStreams geram DataStreams
- Tipo de dado – DataSet
  - Abstração de dados em lote (finito)
  - Operações em Datasets geram Datasets

Apache Flink - Dados

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- API DataStream e API DataSet
  - Fornecem funções operadores para tratar dados distribuídos
  - Extensão do MapReduce com novos operadores nativos
    - Join, Cross, Union, Iterate, Iterate delta, cogroup*, filter...*
  - Permitem definir parâmetro de paralelismo para cada função
    - Instâncias paralelas são atribuídas aos *slots* de processamento
  - Operadores iniciam a execução em memória
    - Em caso de sobrecarga, executam fora de memória

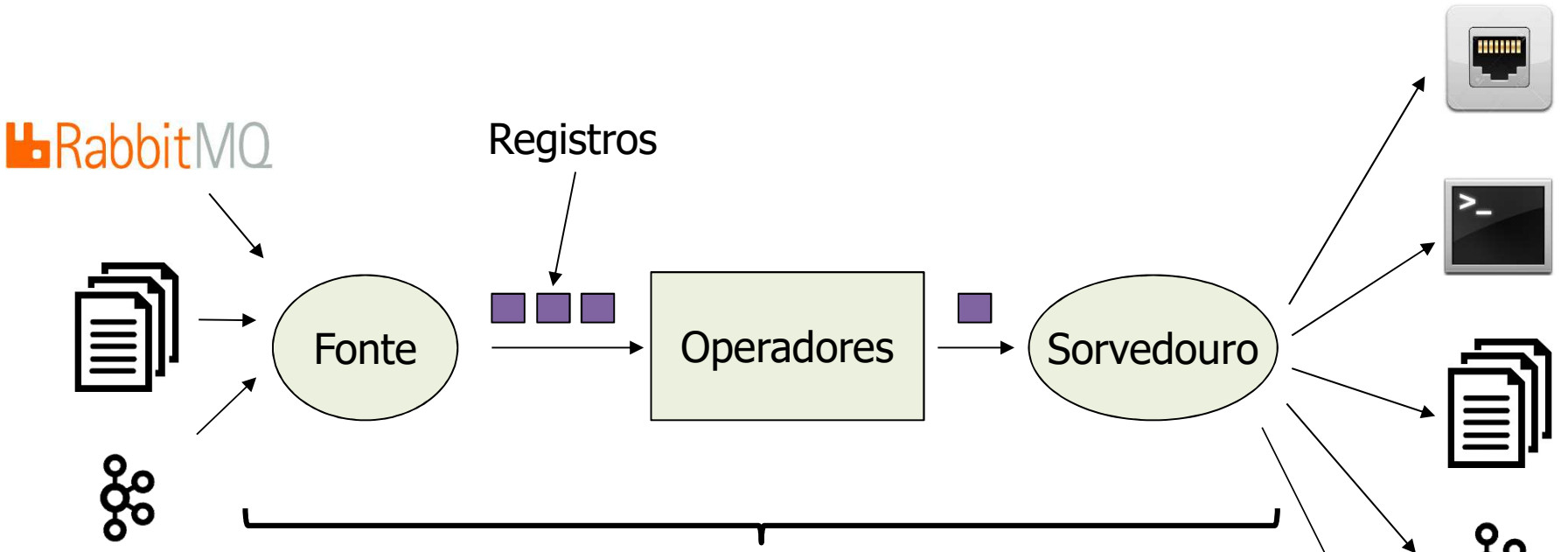
Apache Flink - Integração com Ferramentas Abertas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Service	Open Source Tool
Storage/Serving Layer	HDFS, APACHE HBASE, mongoDB, redis
Data Formats	JSON JavaScript Object Notation, Parquet, AVRO
Data Ingestion Services	kafka, FLUME, RabbitMQ
Resource Management	hadoop

COPPE UFRJ

Apache Flink - Topologia




```
DataStream<String> lines = env.addSource(
                                new FlinkKafkaConsumer<>(...));

DataStream<Event> events = lines.map((line) -> parse(line));

DataStream<Statistics> stats = events
        .keyBy("id")
        .timeWindow(Time.seconds(10))
        .apply(new MyWindowAggregationFunction());

stats.addSink(new RollingSink(path));
```

Source

Transformation

Transformation

Sink

Apache Flink - Topologia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

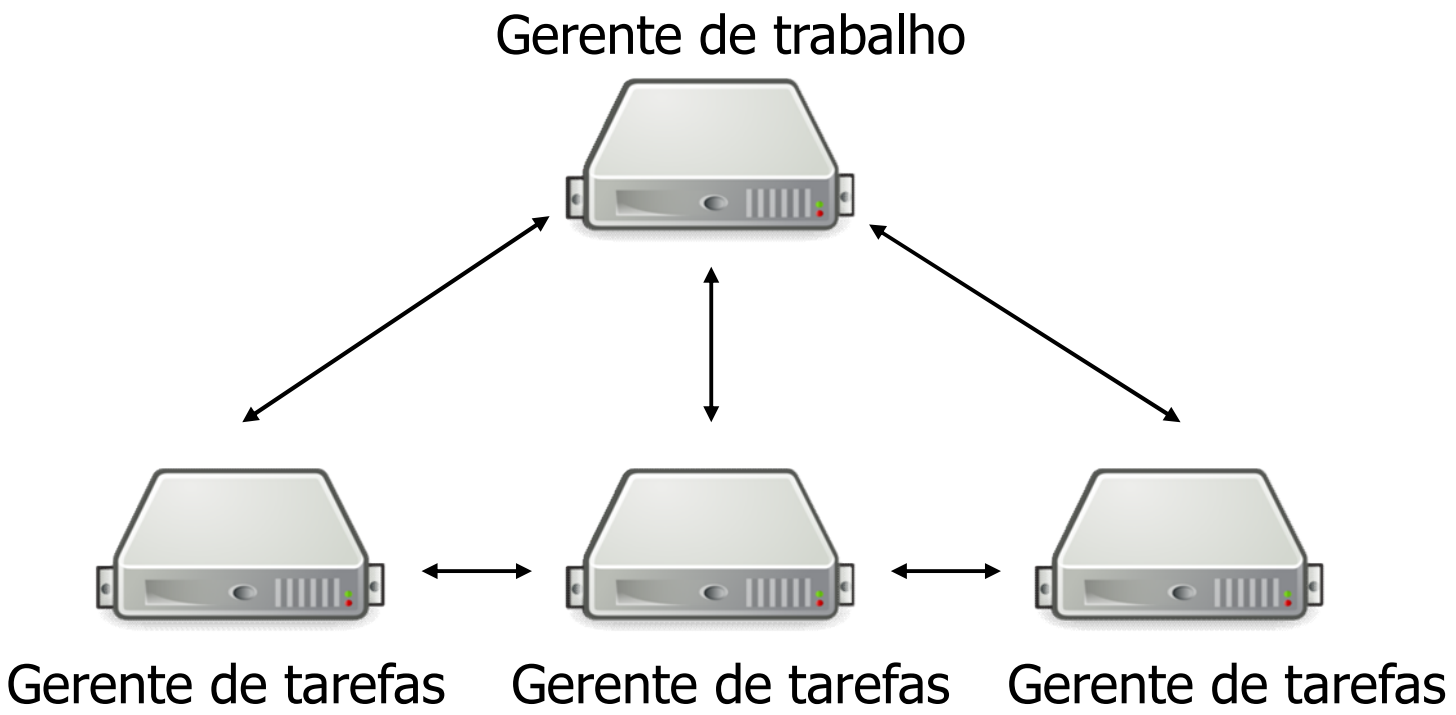
- Topologia abstraída em fluxo de dados cíclicos
  - Permite iterações
- Elementos da topologia
  - Fontes de dados
    - Sistema de mensagens (Kafka), leitura de arquivo texto
  - Operadores
    - Executa o programa/processamento sobre o fluxo de dados
  - Sorvedouro (*sink*)
    - Redireciona o fluxo de dados para um arquivo, *socket*, sistema externo ou imprime na tela
  - Registros
    - Unidade atômica do fluxo de dados

COPPE UFRJ

Apache Flink - Processamento

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Mestre



Trabalhadores

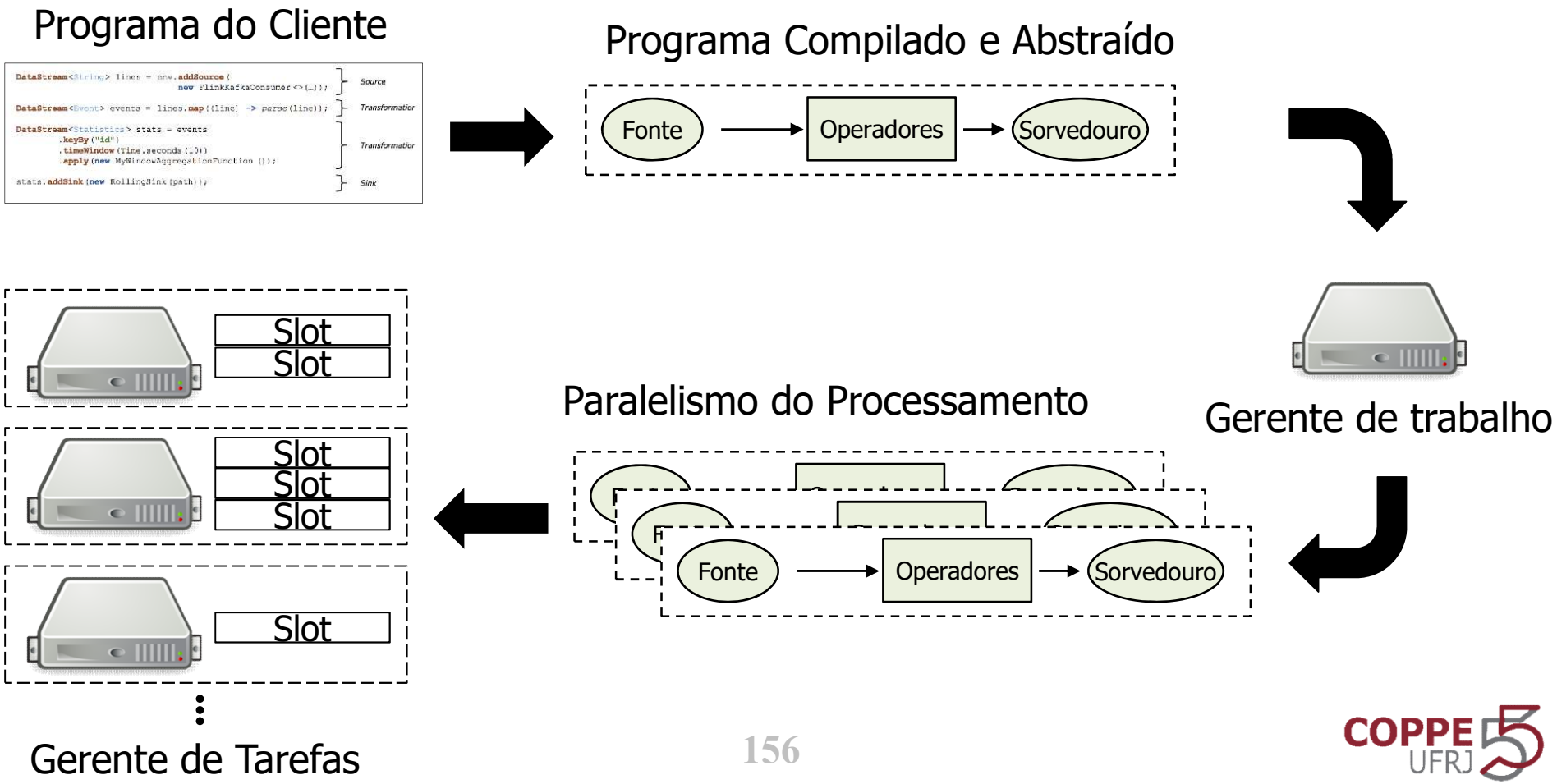
COPPE UFRJ

Apache Flink - Processamento

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

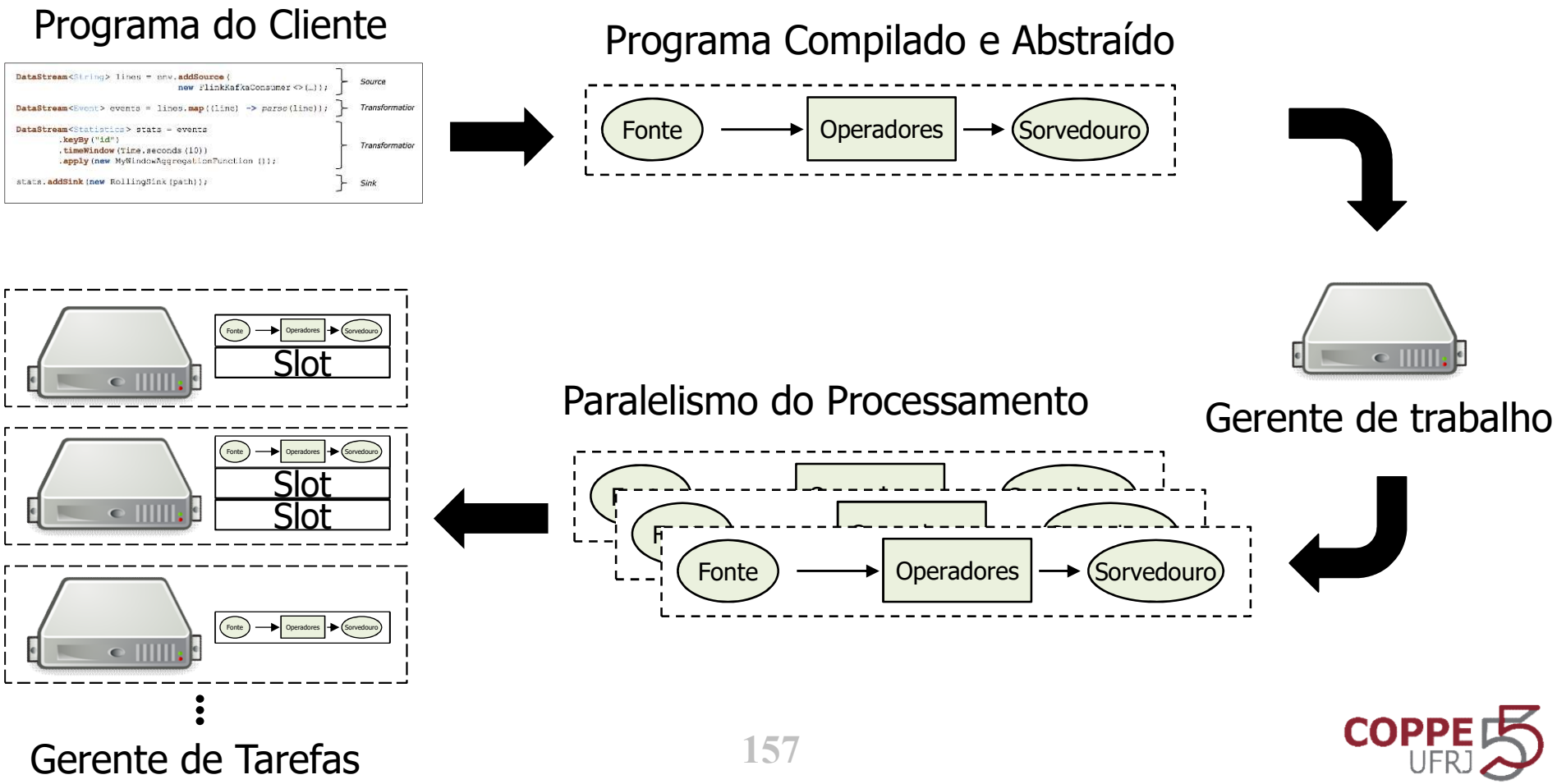
- Nó mestre – **Gerente de trabalho**
  - Recebe as aplicações do cliente
  - Organiza as aplicações em tarefas
  - Envia as tarefas para os gerente de tarefas
  - Mantém os estados das execuções dos gerentes de tarefas
- Nós trabalhadores – **Gerentes de tarefas**
  - Executam tarefas específicas atribuídas pelo nó mestre
  - Utilizam o mecanismo *heartbeat* para informar o estado ao nó mestre
  - Disponibiliza *slots* de processamento para o aglomerado
    - Permite executar tarefas em paralelo

Apache Flink - Processamento



Apache Flink - Processamento

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

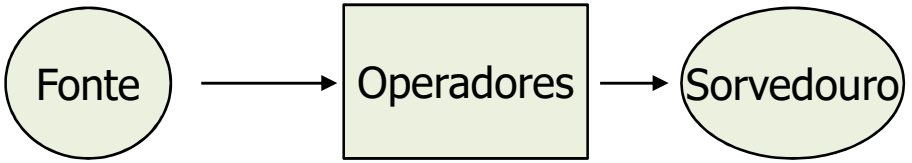


COPPE UFRJ

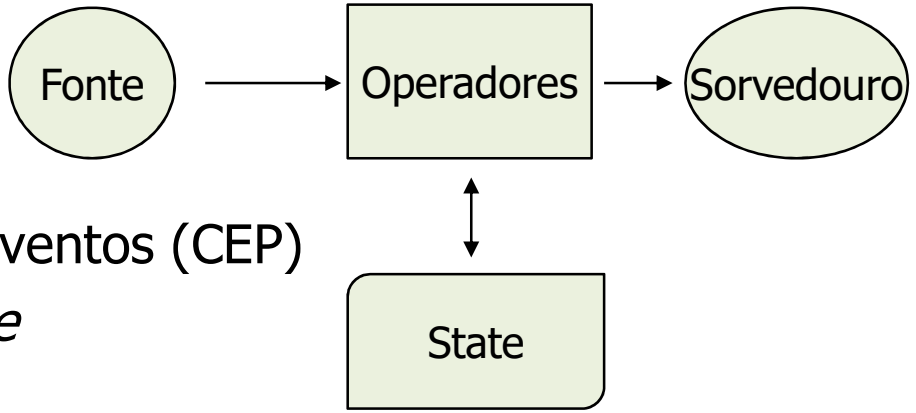
Apache Flink – Sem estados vs. com estados

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Execução sem estados**
  - Consumo de dados
  - Limpeza de dados
  - Transformação sem estados



- **Execução com estados**
  - Agregação
  - Contadores
  - Processamento Complexo de Eventos (CEP)
  - Aprendizado de máquina *online*



COPPE UFRJ

Apache Flink – Sem estados vs. com estados

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

O que acontece se for preciso...

- Mudar número de trabalhadores
- Migrar para um novo aglomerado
- Corrigir um bug no código
- Atualizar versão do Flink
- Testar diferentes versões de algoritmos

- **Execução sem estados**
  - Fácil de operar e manipular
  - Basta parar e recomeçar o trabalho
- **Execução com estados**
  - Não trivial → Requer artifícios extras
    - Estado precisa ser recarregado e redistribuído
  - Uso de *savepoints* → Armazenar estado externamente

COPPE UFRJ

A problem has been detected and Windows has been shut down to prevent damage to your computer.

PFN_LIST_CORRUPT

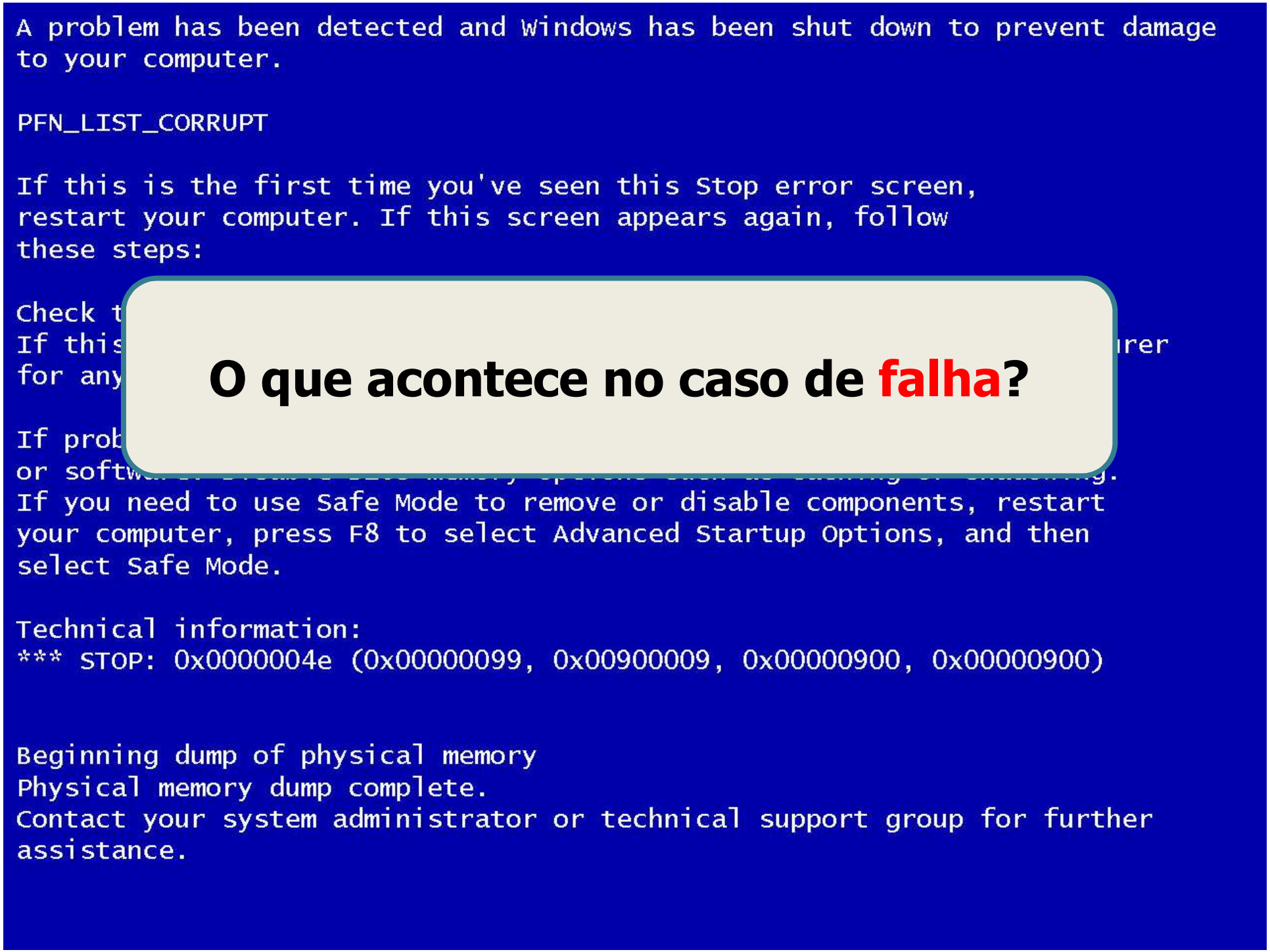
If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:
*** STOP: 0x0000004e (0x00000099, 0x00900009, 0x00000900, 0x00000900)

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further assistance.

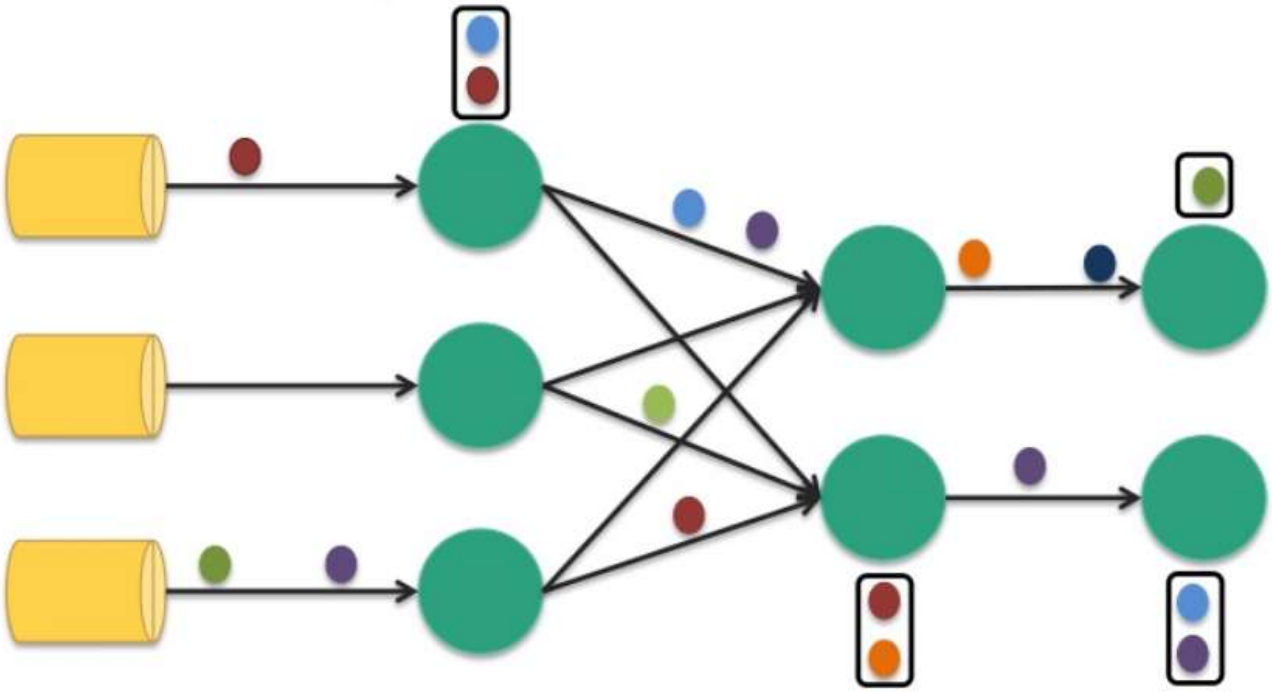


O que acontece no caso de falha?

Apache Flink – Tolerância a Falhas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

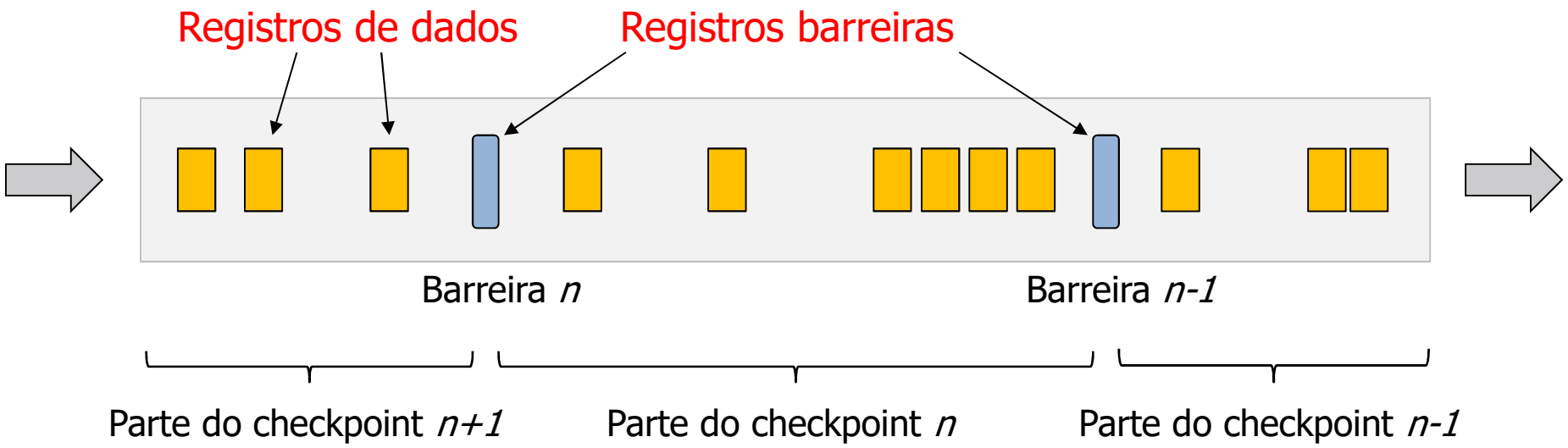
- *Checkpoint* → *Snapshots* consistentes dos fluxos de dados distribuídos e dos estados dos operadores



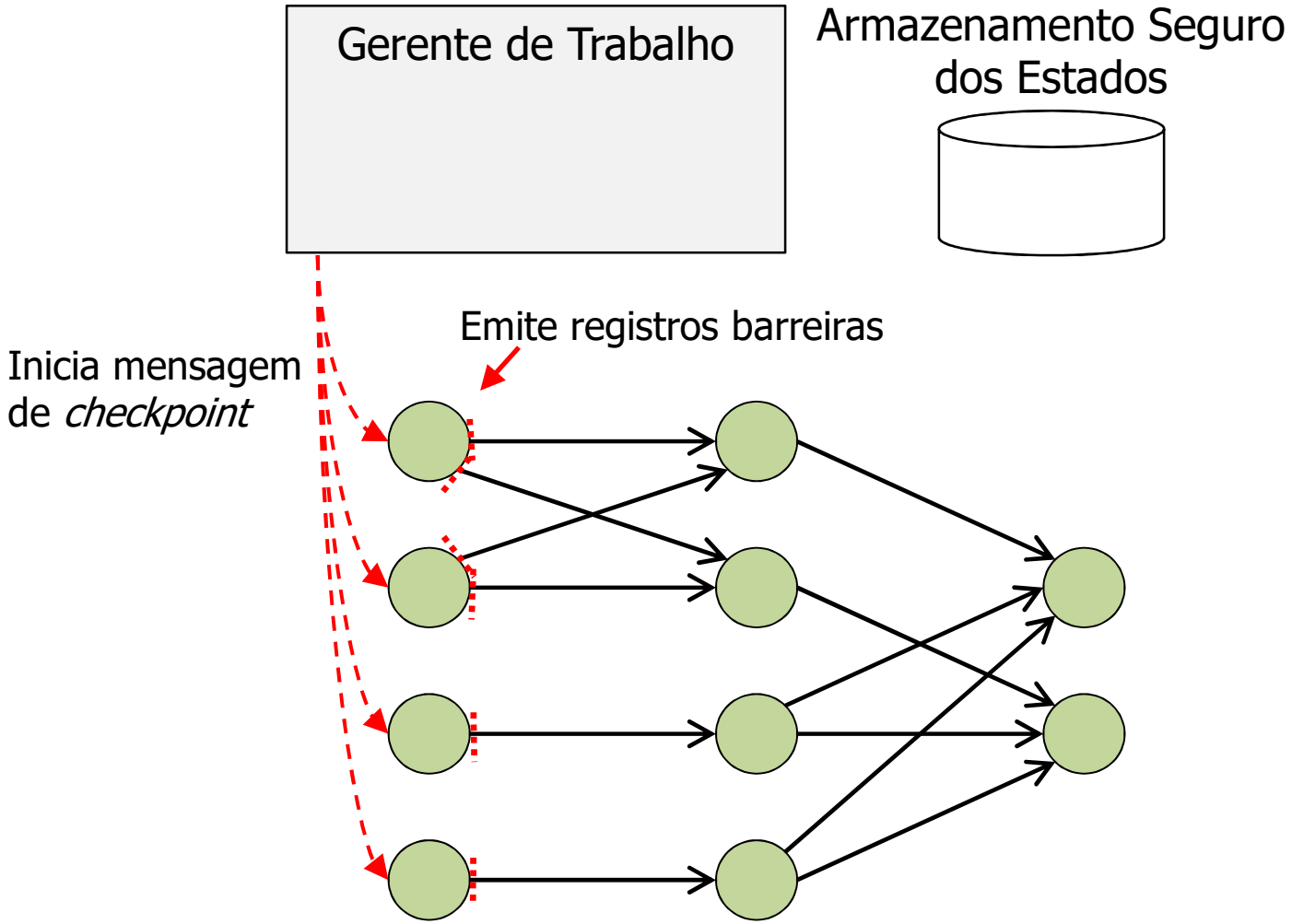
COPPE UFRJ

Apache Flink – Tolerância a Falhas

- Barreiras → Marcadores para *checkpoints*
  - Injetados no fluxo de dados em forma de registros

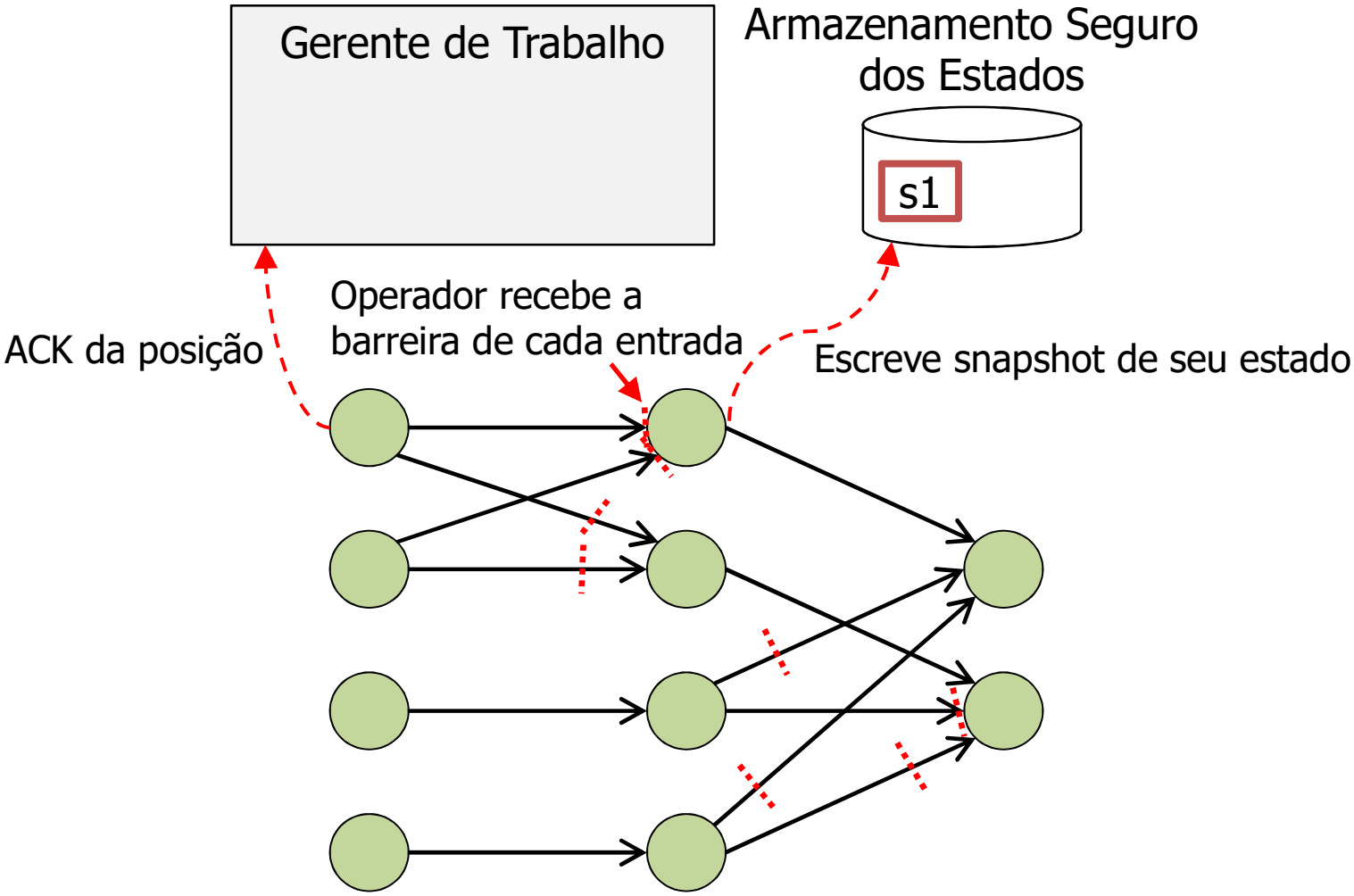


Apache Flink – Tolerância a Falhas



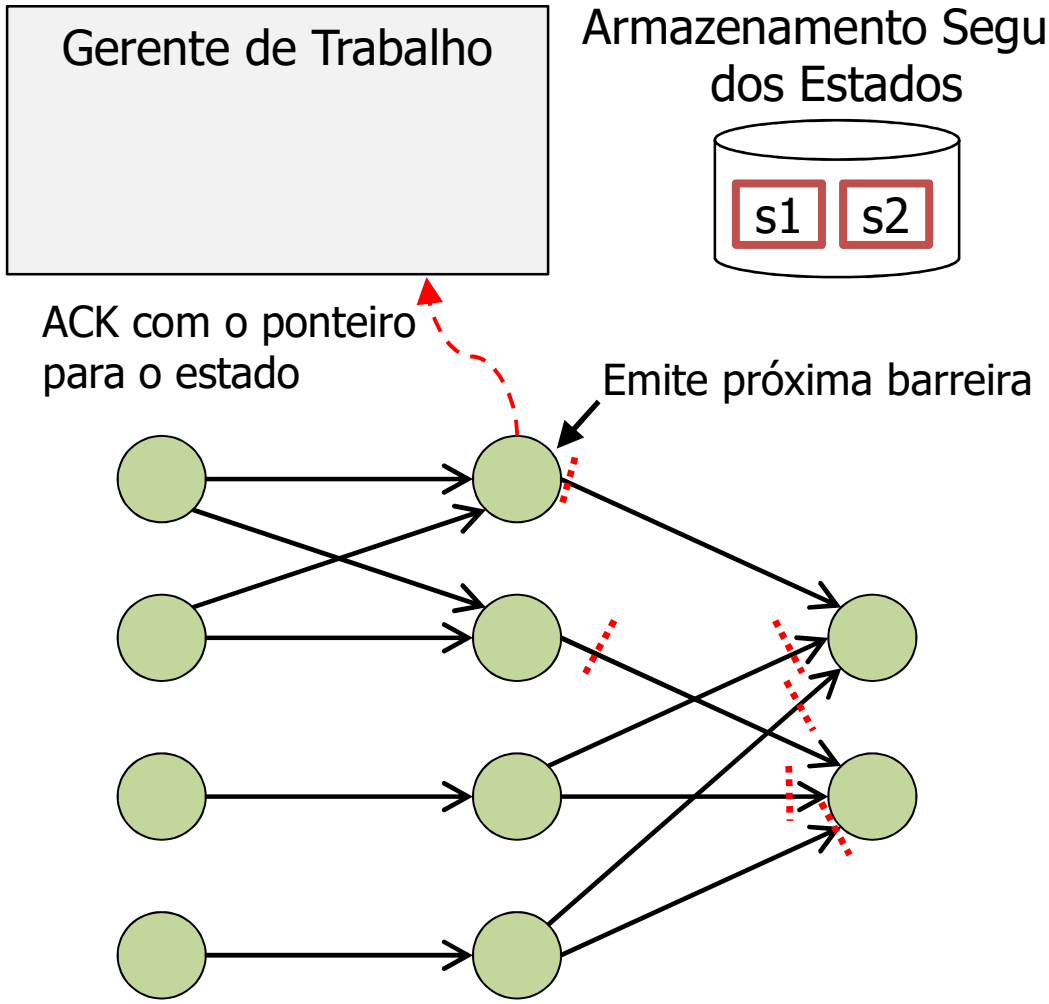
Iniciando Checkpoint

Apache Flink – Tolerância a Falhas



Checkpoint em Progresso

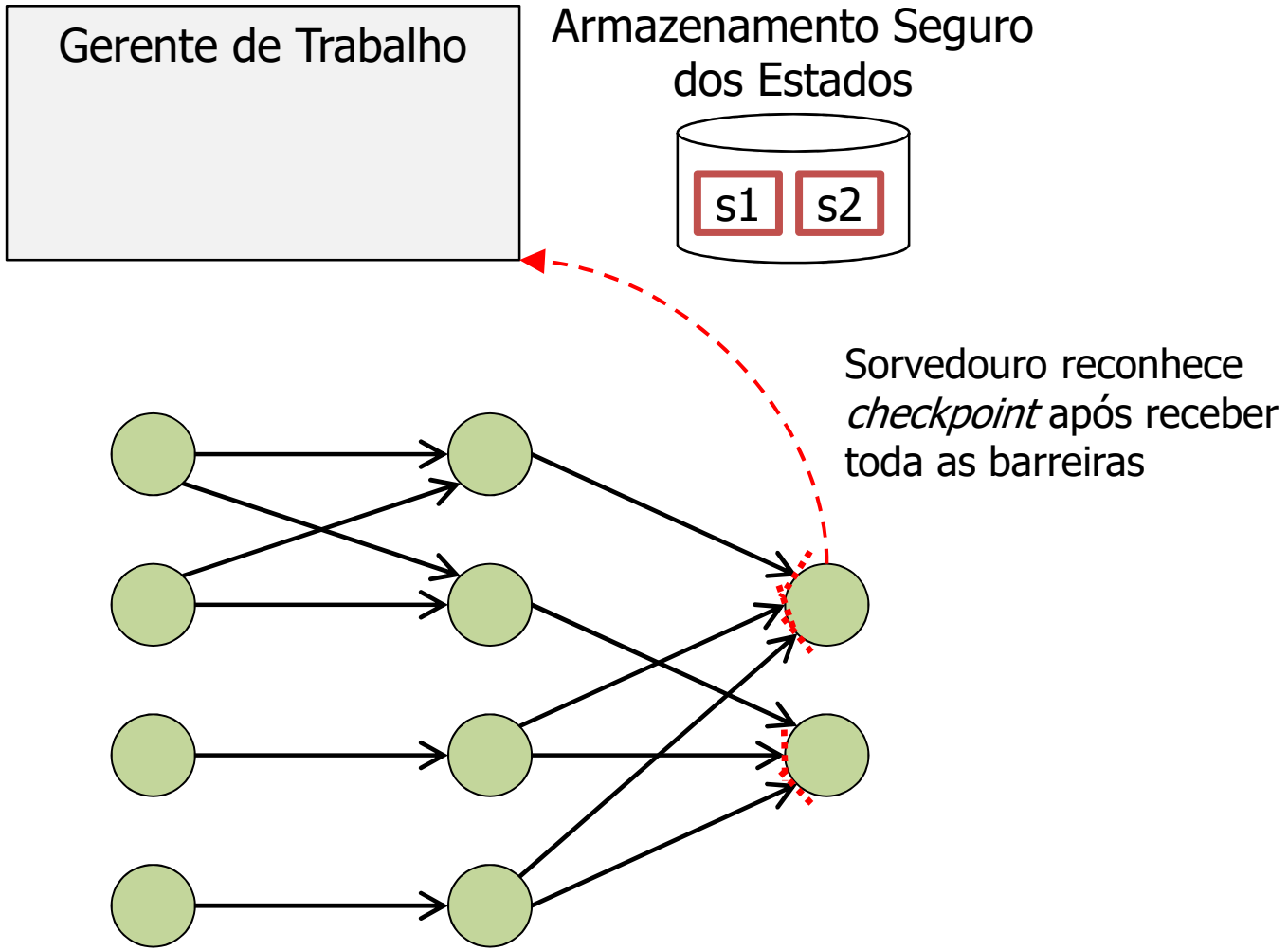
Apache Flink – Tolerância a Falhas



Checkpoint em Progresso

Apache Flink – Tolerância a Falhas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Checkpoint Completo

COPPE UFRJ

Apache Flink - Semânticas de Operação

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Modos de operação
  - Semântica "no máximo uma"
    - Sem garantia
  - Semântica "no mínimo uma"
    - Suficiente para maioria das aplicações
  - Semântica "exatamente-uma"
    - Mecanismo de tolerância a falhas com *snapshots e checkpoints*
    - Supõe-se que Flink é precedido por um sistema de mensagens de reenvio persistente
      - Apache Kafka, RabbitMQ...
    - Supõe que Flink possui Alta Disponibilidade (HA)
      - Apache Zookeeper → Gerentes de Trabalho redundantes

Apache Flink - Semânticas de Operação

- Modos de operação
  - Semântica "no máximo uma"
    - Sem garantia
  - Semântica "no mínimo uma"
    - Suficiente para maioria das aplicações
  - Semântica "exatamente-uma"
    - Mecanismo de tolerância a falhas com *snapshots e checkpoints*
    - Supõe-se que Flink é precedido por um sistema de mensagens de reenvio persistente

Processamento em lotes não tolera falhas

→ Operação deve ser reiniciada do zero

COMPARAÇÃO ENTRE APACHE SPARK STREAMING APACHE STORM APACHE FLINK

Comparação entre as Plataformas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Processamento
  - Storm → Processamento por fluxo
    - Storm Trident
  - Flink → Processamento por fluxos e em microlotes nativo
  - Spark → Processamento em lotes
    - Spark Streaming → em tempo quase-real por microlotes

COPPE UFRJ

Comparação entre as Plataformas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Abstração de dados
  - Spark → DStream e Dataframe estão em cima do RDD
  - Flink → DataStream e Dataset são abstrações independentes
  - Storm → Tuplas (Somente fluxo de dados)
- Linguagem
  - Flink → Escrito em Java e suporta Java/Scala
  - Spark → Escrito em Scala e suporta Java/Scala/Python
  - Storm → Escrito em Java e suporta "qualquer linguagem"

Comparação entre as Plataformas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Gerenciamento de memória
  - Spark até 1.4 → Java Heap para *caching* de dados
  - Spark a partir da 1.5, Storm e Flink → Gerenciamento customizável de memória
- Processamento Iterativo (Aprendizado de máquina)
  - Spark → Realizado através de vários grafos acíclicos (DAG)
    - Agenda e executa novo conjunto de tarefas a cada iteração
  - Storm →Não apresenta suporte nativo
    - Tornado → Modelo de programação paralela de grafos
  - Flink → Suporta grafos cíclicos nativamente
    - De forma controlada e em tempo de execução
    - Melhor escalabilidade e desempenho do que DAGs

COPPE UFRJ

Comparação entre as Plataformas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Tolerância a falhas
  - Flink → Exatamente 1 vez nativo
  - Spark → Exatamente 1 vez através de Spark Streaming
  - Storm → Pelo menos 1 vez através do Storm Trident

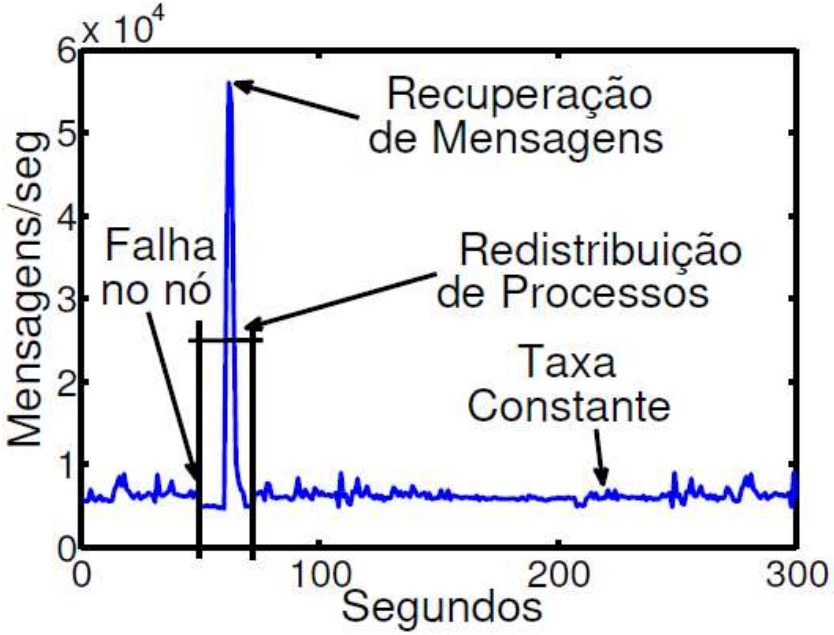
- Mecanismos de tolerância a falhas
  - Flink → Barreiras e Checkpoints
  - Spark → Recuperação
  - Storm → Backup e reenvio através de acks

COPPE UFRJ

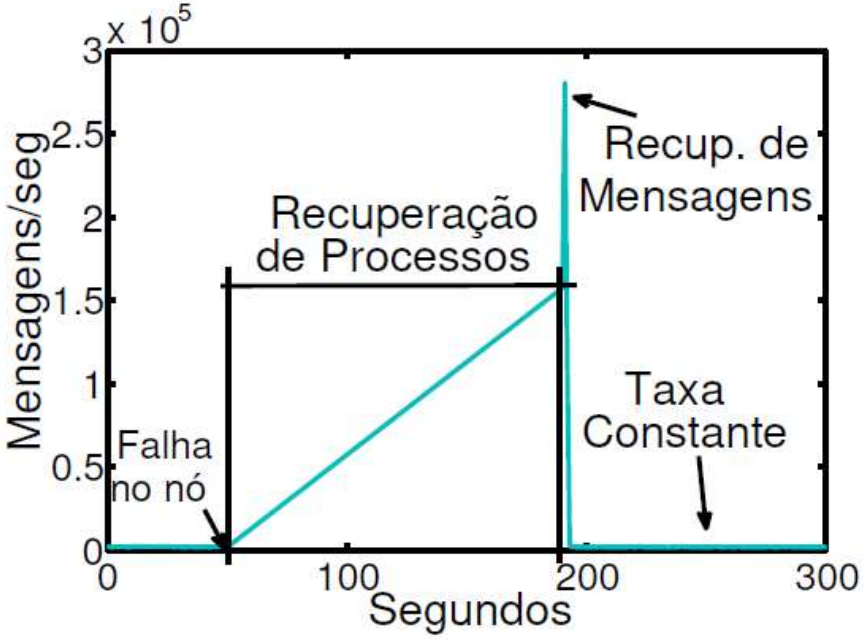
Perda de Mensagens durante a Falha

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Storm



Flink

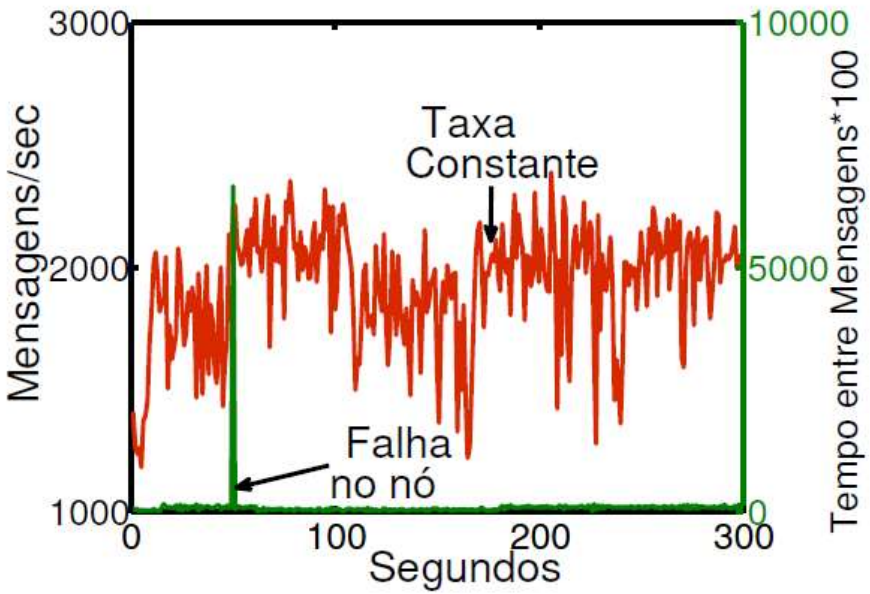


COPPE UFRJ

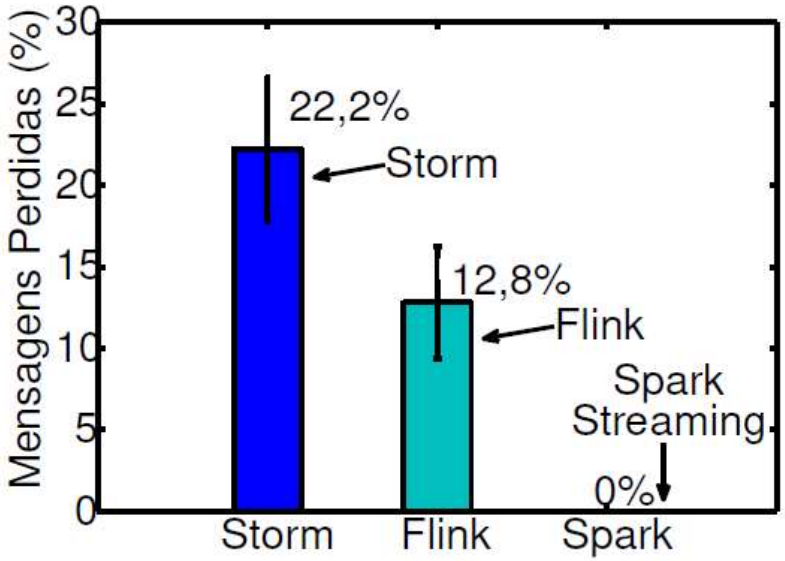
Perda de Mensagens durante a Falha

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Spark



Storm vs. Flink vs. Spark



COPPE UFRJ

PARTE III

Detecção de Ameaças usando Aprendizado de Máquina

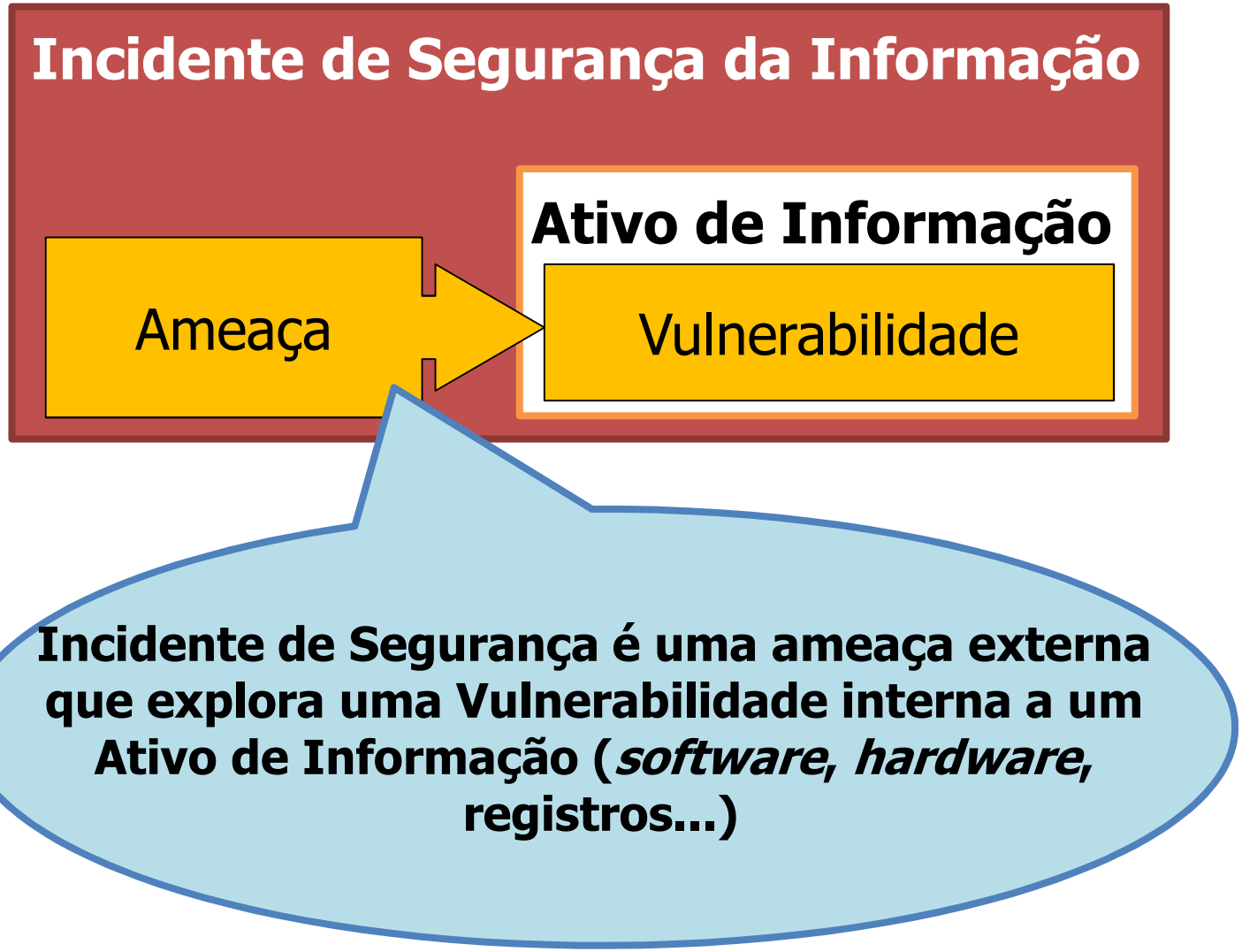
Ameaças, riscos, impactos e a detecção em tempo real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Incidente de Segurança da Informação

Ameaças, riscos, impactos e a detecção em tempo real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

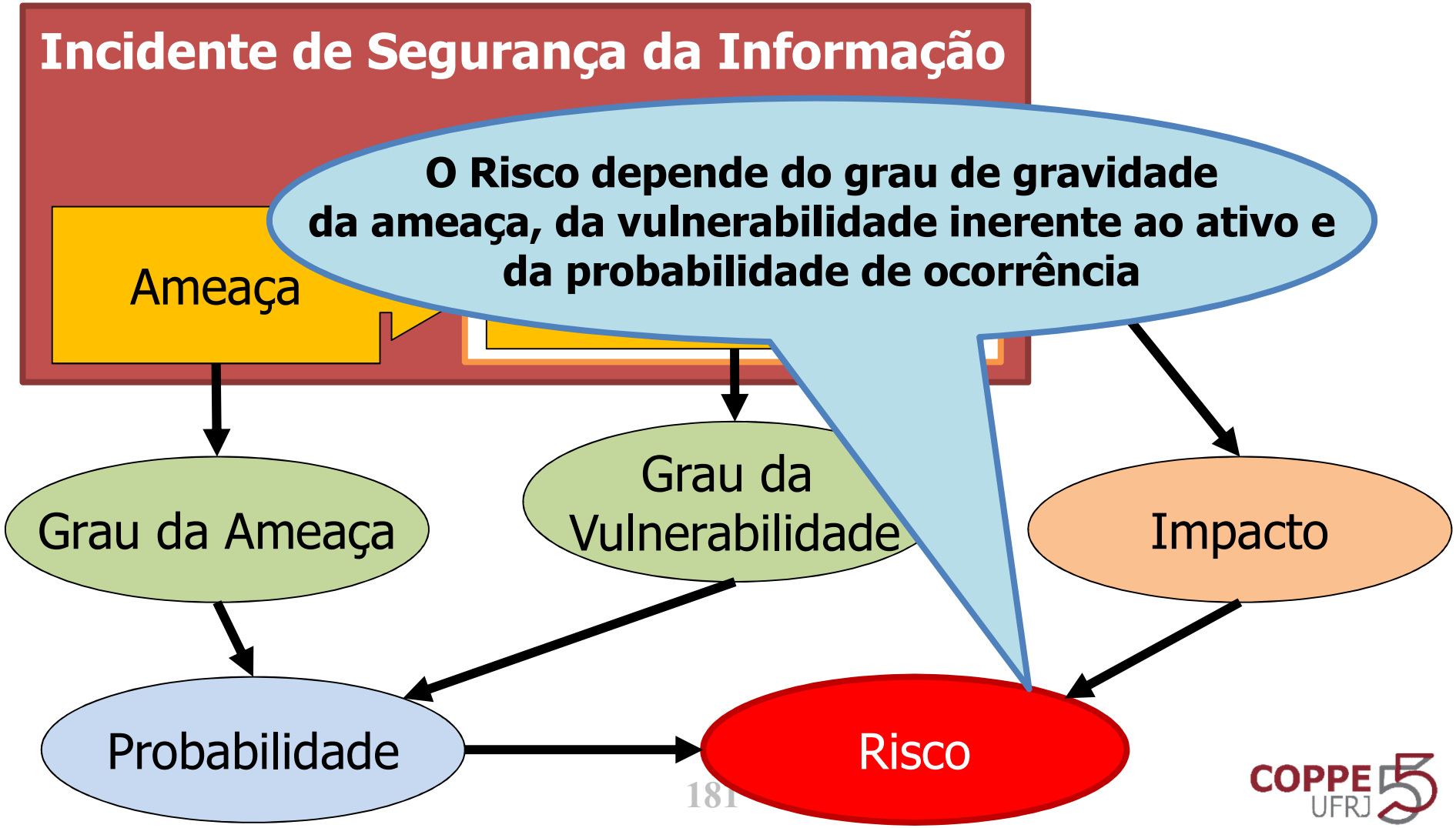
Ameaças, riscos, impactos e a detecção em tempo real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

COPPE UFRJ

Ameaças, riscos, impactos e a detecção em tempo real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Ameaças, riscos, impactos e a detecção em tempo real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Sistemas atuais demoram até semanas para identificar a ocorrência de um

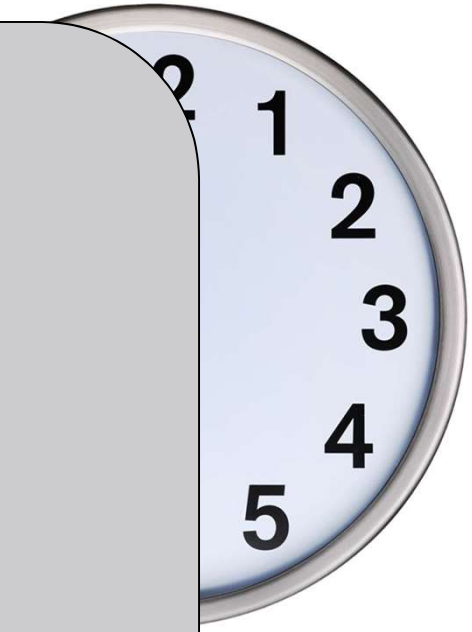
Incidente de Segurança

Tarde demais para limitar os IMPACTOS

COPPE UFRJ

Ameaças, riscos, impactos e a detecção em tempo real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Proposta de solução

Emprego de técnicas de **aprendizado de máquina** para **automatizar a detecção** e garantir a **detecção rápida**

Tarde demais para limitar os IMPACTOS

COPPE UFRJ

A detecão de ameaças

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Monitoramento de tráfego
  - Captura dos pacotes
  - Análise dos cabeçalhos
  - Separação em fluxos
  - Análise do comportamento do fluxo
    - Verificação de assinaturas
    - Detecção de anomalias
  - Classificaficação do fluxo
    - Normal
    - Malicioso
  - Geração de alarmes
  - Reação a ataques

O Problema da Detecção de Ameaças em Tempo Real

Problema: classificação dos fluxos de dados assim que amostras do fluxos chegam ao sistema de classificação

- **Desafios**
  - Classificação de dados de fluxo potencialmente ilimitados
  - Tomada de decisão limitada a tempos próximo ao tempo real
    - Cálculos complexos por amostra são impraticáveis
  - Escolha do conjunto de dados para treino adequado
  - Escolha do algoritmo de classificação, das características de representação dos dados,
  - Definição da arquitetura de detecção em fluxo
  - Avaliação e comparação de classificadores

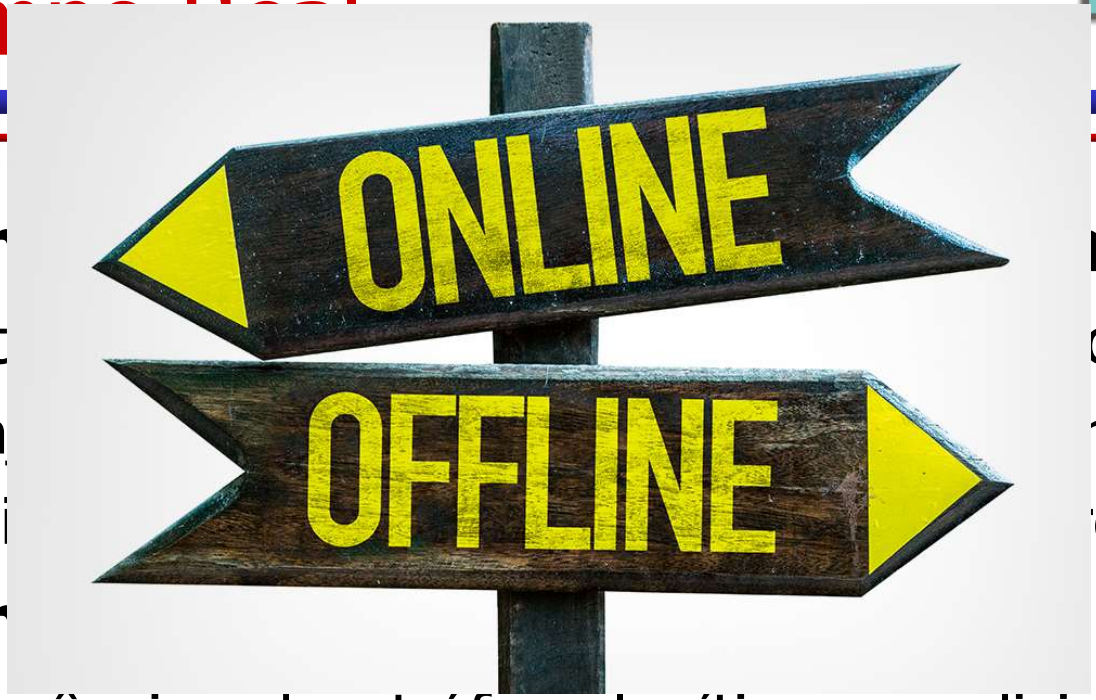
Detecção de Ameaças em Tempo Real

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Treinamento em tempo diferenciado (*offline*)**
  - detecção de ameaças previamente conhecidas
  - conjunto de dados disponível marcados em normal e ameaça
  - aplicação de modelo treinado em tempo real
- **Treinamento com dados em linha (*online*)**
  - **i**nferência sobre tráfego legítimo e malicioso
  - detecção de ameaças em tempo real
  - detecção de ameaças inéditas
    - *Zero day attacks* → novas ameaças que exploram vulnerabilidades até então desconhecidas

Detecção de Ameaças em Tem

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



- **Treir** **ndo (*offline*)**
  - det cidas
  - con normal e ameaça
  - apli eal
- **Treir** **(*online*)**
  - **i**nferência sobre tráfego legítimo e malicioso
  - c
  - c

ram

Quais os conjuntos de dados disponíveis para treino ?
Qual é o conjunto de dados mais adequado para treino?

COPPE UFRJ

Avaliação das Métodos de Segurança

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Desafio:** obter um conjunto de dados adequado para as avaliações
- Disponibilidade de conjuntos de dados na literatura é limitada
  - Preocupação com a privacidade
  - Receio de vazamento de informações confidenciais contidas na carga útil de pacotes

Conjunto de Dados de Segurança

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Tabela 3.3. Resumo dos principais conjuntos de dados disponíveis na literatura (Ca: características; Fl: fluxos).

Conjunto	Formato	Tamanho	Ataques	Tipo	Ano
DARPA 98/99	pcap	9.87 GB	80.1 %	Sintético	1998/99
KDD99	csv (41 Ca)	805.050 Fl	80.1 %	Sintético	1998
NLS-KDD	csv (41 Ca)	148.517 Fl	80.5 %	Sintético	1998
LBNL	pcap	11 GB	-	Real	2005
Kyoto	txt (24 Ca)	14 GB	100%	Real	2006
CAIDA DDoS	pcap	21 GB	100%	Real	2007
ISCX IDS	pcap	84.42 GB	2.8 %	Sintético	2012
CTU-13	pcap	697 GB	11.69 %	Sintético	2014
MAWI	pcap	~ 2 GB/ano	-	Real	2001–18
GTA/UFRJ	csv (26 Ca)	95 GB	30 %	Real/Controlado	2016
NetOp	csv (46 Ca)	5.320.955 Fl	-	Real	2017

COPPE UFRJ

Conjunto de Dados de Segurança

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

DoS, Probe, User2Root (U2R), Remote2Local (R2L)

Malware de honeypot

DDoS

Trafego de Botnet

Tabela 3.3. F… dados disponíveis na literatura
(Ca: caracter…

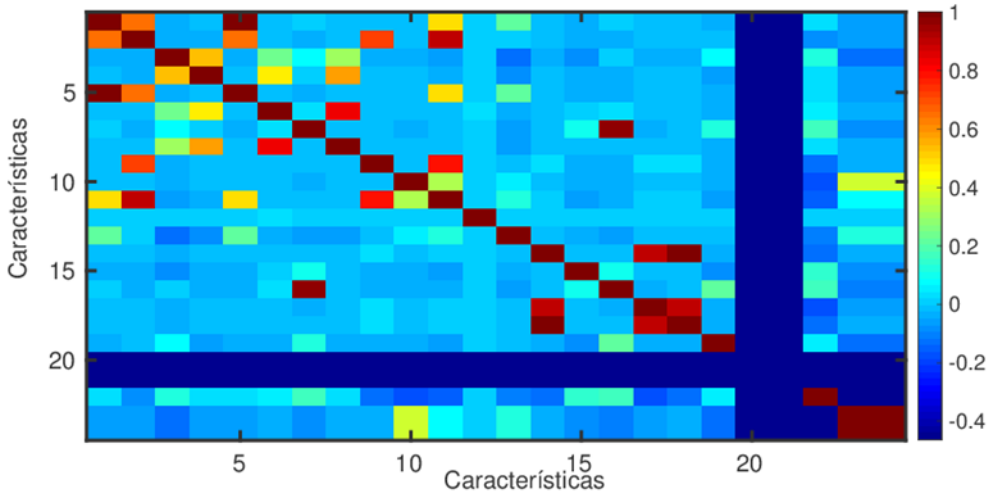
Conjunto			Ataques	Tipo	Ano
DARPA 98/99	pcap	…8/ GB	80.1 %	Sintético	1998/99
KDD99	csv (…	…Fl	80.1 %	Sintético	1998
NLS-KDD	csv…	…l		Sintético	1998
LBNL	p…	…		Real	2005
Kyoto	txt (24 Ca)	14 GB		Real	2006
CAIDA DDoS	pcap	…		Real	2007
ISCX IDS	p…	84.42 GB		Sintético	2012
CTU-13	pcap	697 GB		Sintético	2014
MAWI	pcap	~ 2 GB/a…		Real	2001–18
GTA/UFRJ	csv (26 Ca)	…		Controlado	2016
NetOp	csv (46 Ca)	5.320.9…		Real	2017

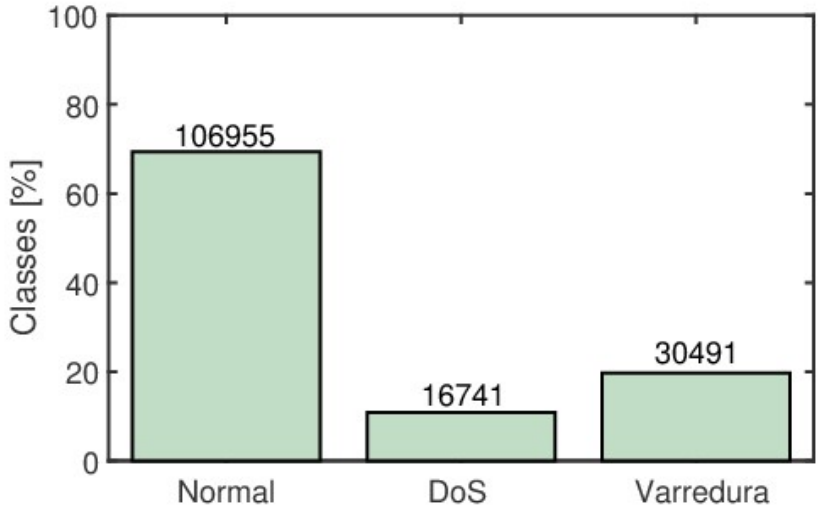
COPPE UFRJ

Conjunto de dados GTA/UFRJ

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- 95 GB de arquivos Pcap
  - Ataques foram criados utilizando a distribuição Kali Linux
    - 7 tipos de DoS, 9 tipos de Varredura (*Scan*)
- 24 características, 3 classes (Normal, DoS, Varredura)
  - 214,200 fluxos





COPPE UFRJ

Conjunto de dados NetOp

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Tráfego de Acesso de 373 usuários de Banda Larga
  - Zona Sul da cidade do Rio de Janeiro
  - 1 semana de dados brutos ininterruptos → *tcpdump*
    - De 24 de fevereiro a 4 de março de 2017
- 5 TB de tráfego real
  - Mais de 5 milhões de fluxos por dia
- 2 classes
  - Normal e Ameaças → anteriormente rotulado pelo Suricata IDS

Seleção de características

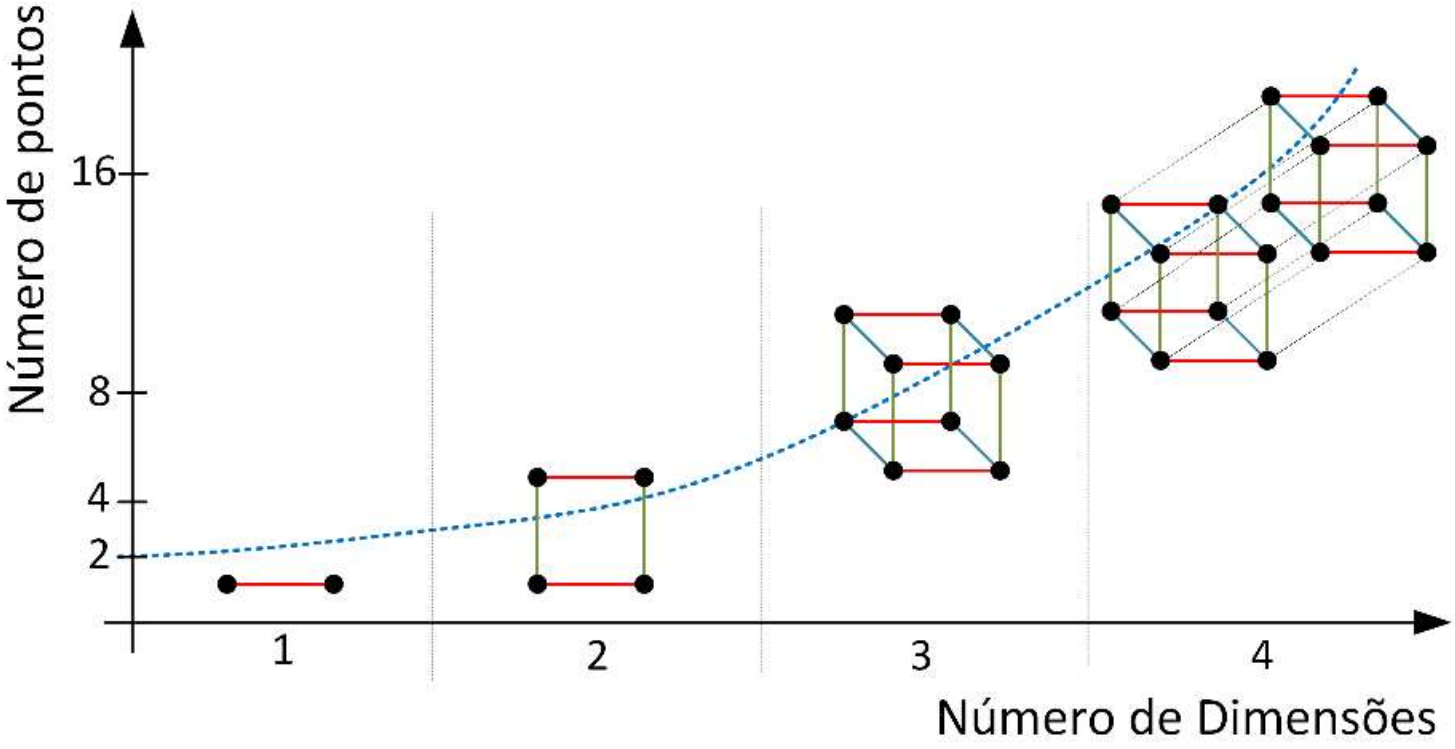
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Objetivo** - escolher um subconjunto ótimo de características de acordo com um determinado critério
- **Consequências**
  - Simplificar o modelo e aumentar a velocidade
  - Aumentar a acurácia e a precisão dos modelos
  - Reduzir a dimensionalidade e remover o ruído
    - Remover dados irrelevantes
    - Possibilitar a visualização dos dados para a seleção do modelo

Maldição da Dimensionalidade

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O número necessário de pontos cresce exponencialmente com o número de variáveis

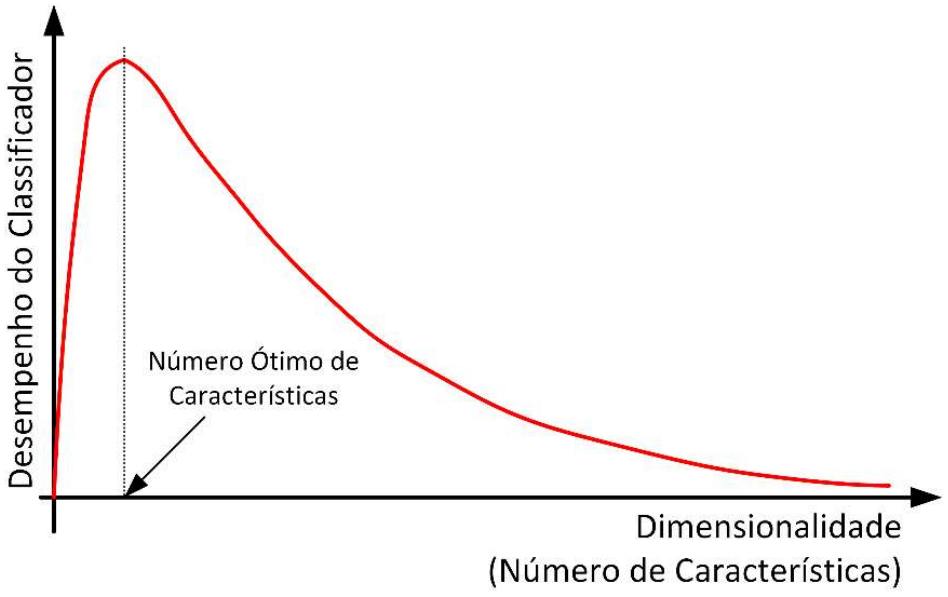


COPPE UFRJ

Maldição da Dimensionalidade

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

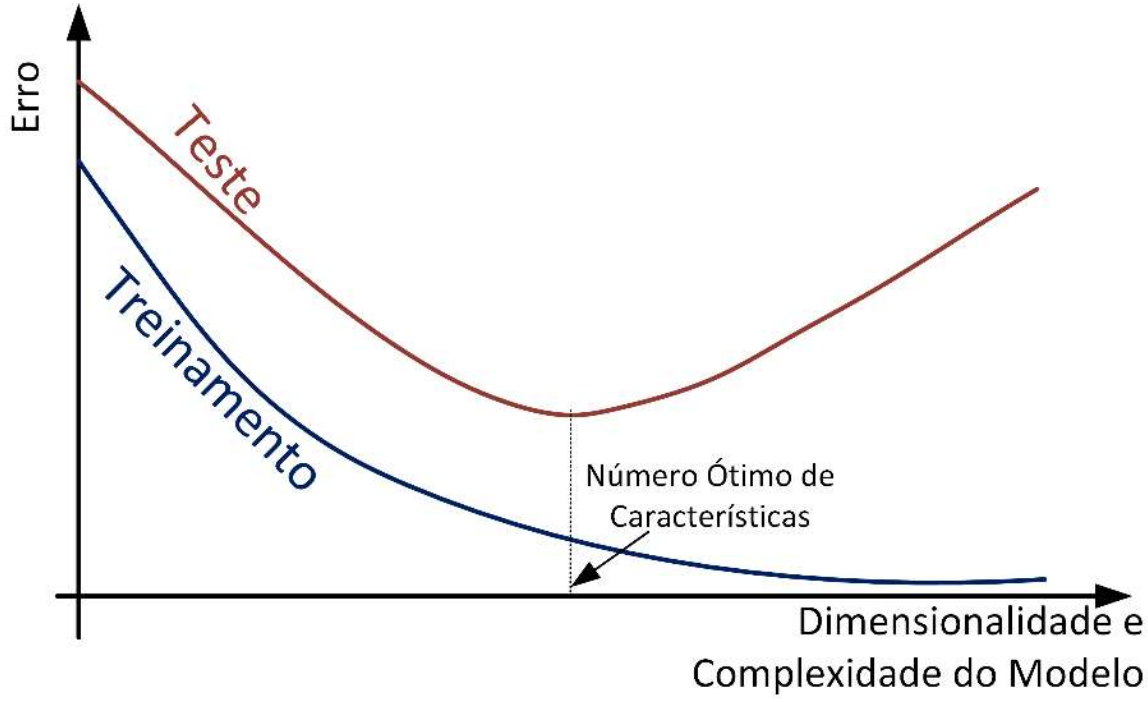
- Na prática: o número de exemplos de treinamento é fixo
- Existe um número ótimo de características
  - A partír do ótimo, aumentar a dimensionalidade resulta em uma diminuição no desempenho do classificador



Maldição da Dimensionalidade

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

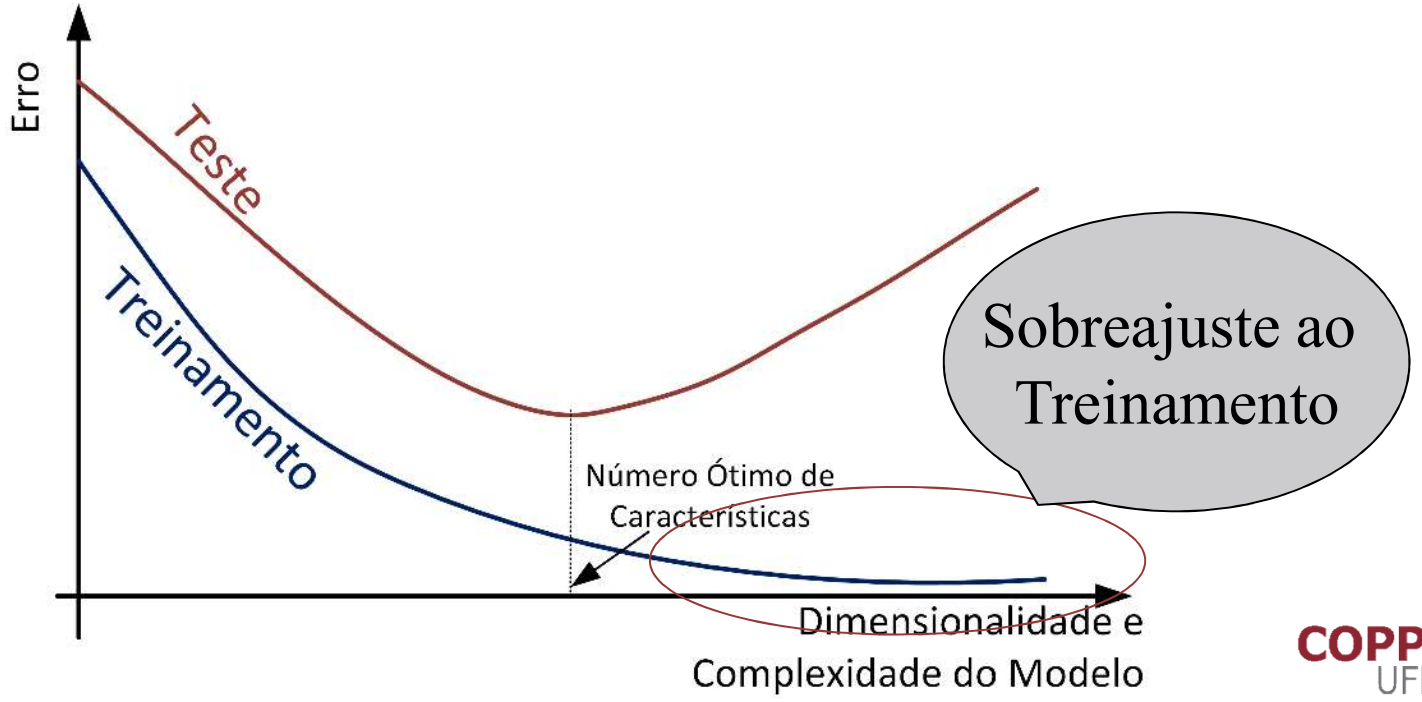
- Na prática: o número de exemplos de treinamento é fixo
- Existe um número ótimo de características
  - A partír do ótimo, aumentar a dimensionalidade resulta em uma diminuição no desempenho do classificador



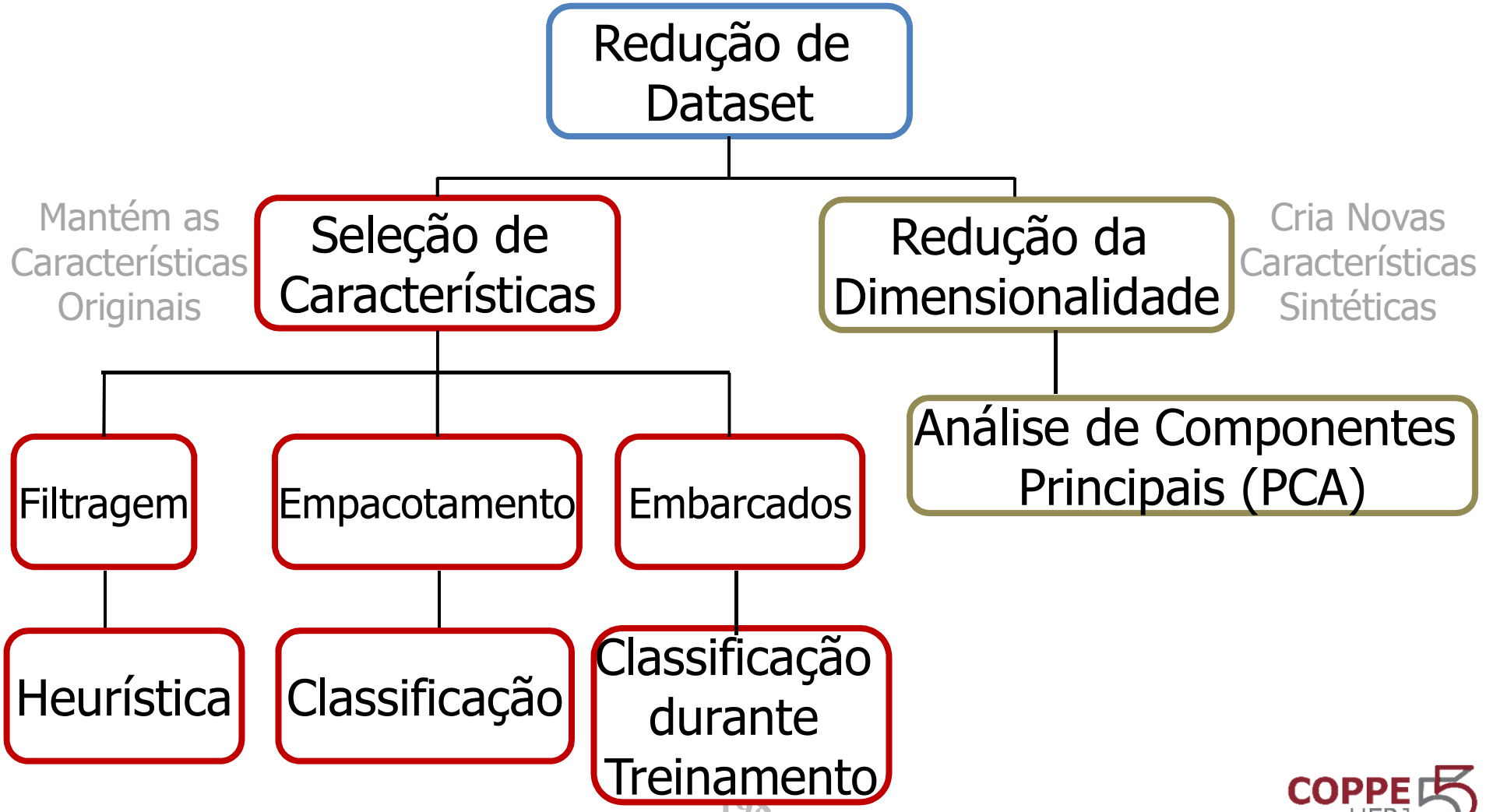
Maldição da Dimensionalidade

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Na prática: o número de exemplos de treinamento é fixo
- Existe um número ótimo de características
  - A partír do ótimo, aumentar a dimensionalidade resulta em uma diminuição no desempenho do classificador

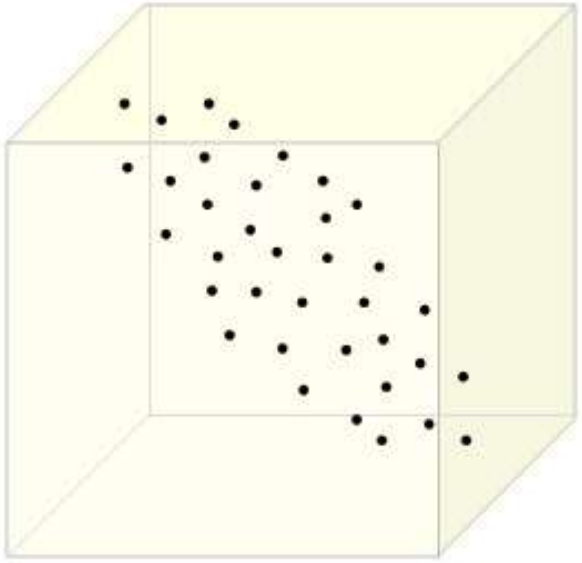


Redução de Conjunto de Dados

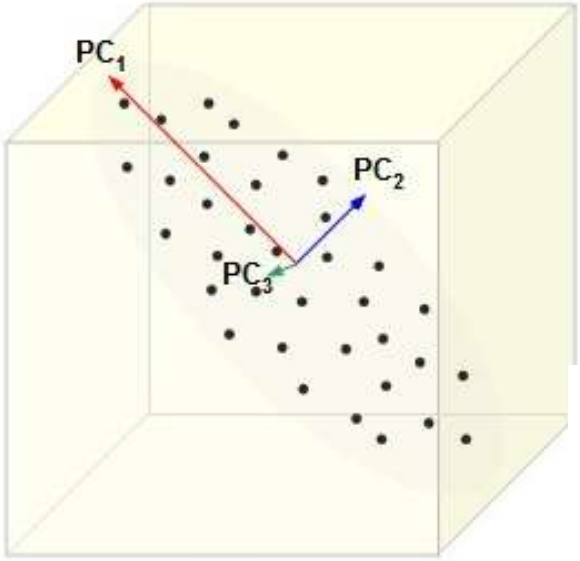


Redução da dimensionalidade
PCA (Linear)

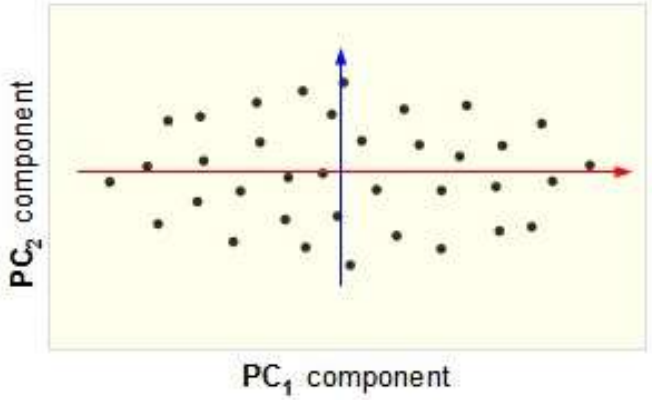
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



a



b

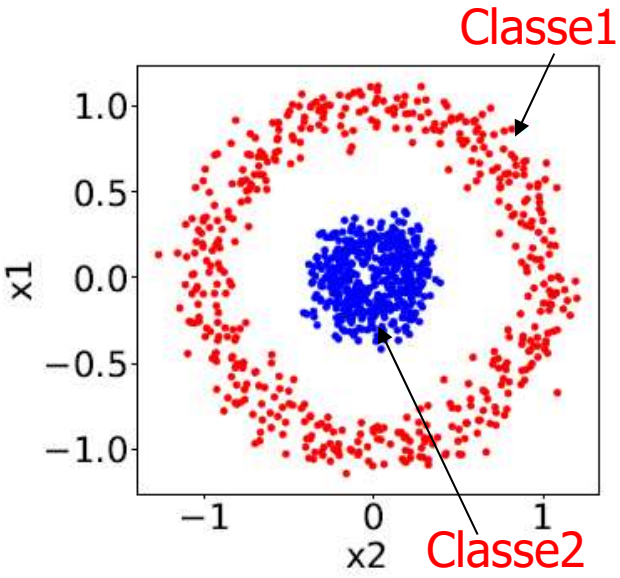


c

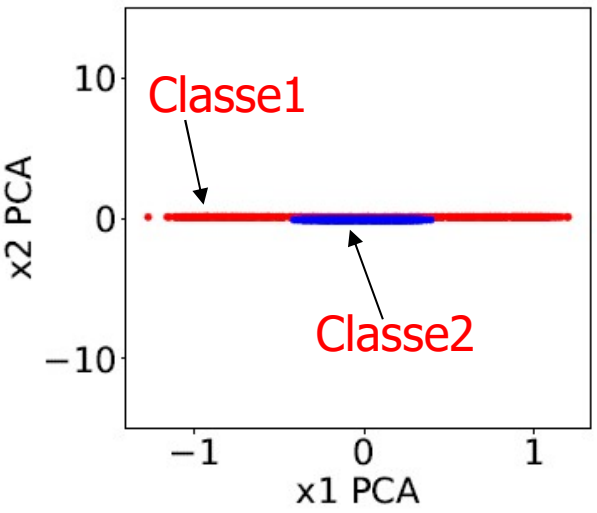
Utilização do PCA em Domínios não lineares

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Domínio Não-linear

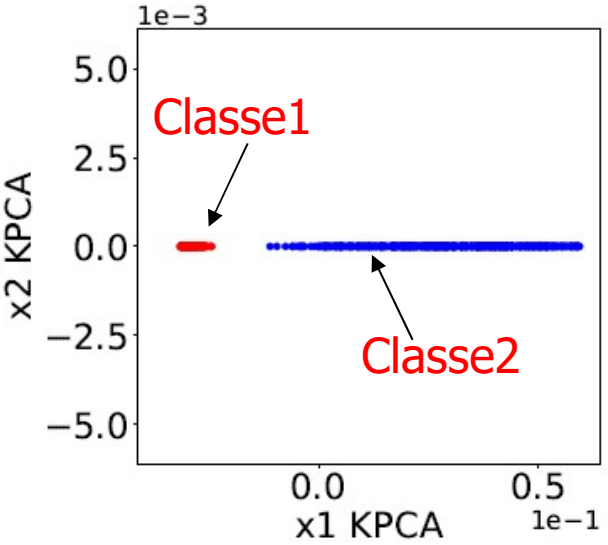


PCA Linear



Separação de Classes não é possível

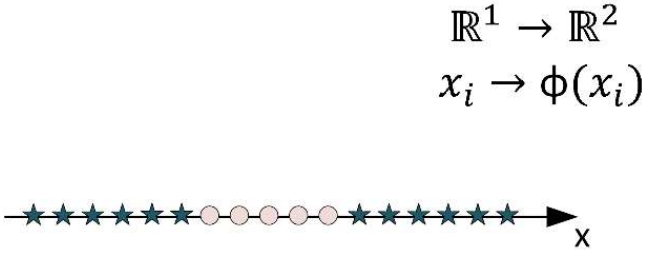
PCA Não-Linear



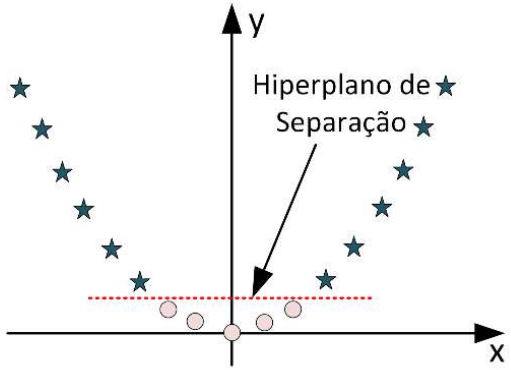
Separação de Classes Possível

COPPE UFRJ

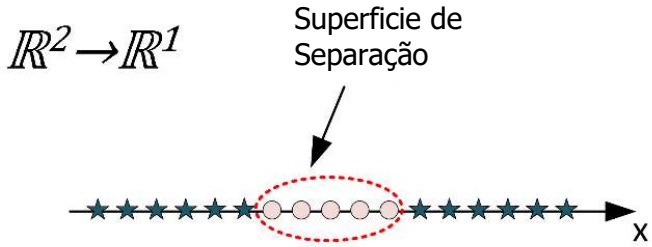
PCA Não-linear (Kernel)



Conjunto Original X



Mapeamento X para um espaço de dimensão superior, e realização de PCA LINEAR nesse espaço



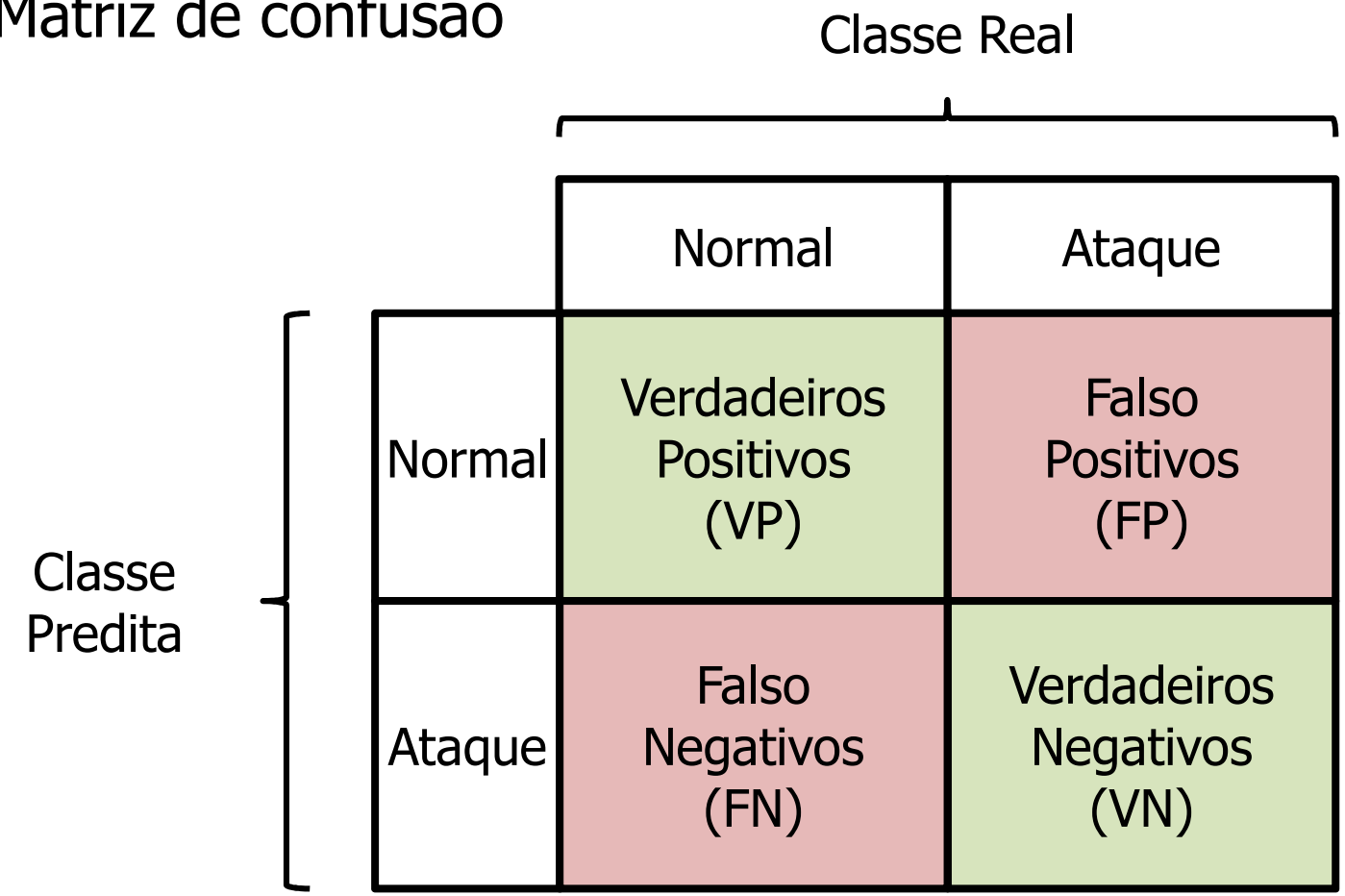
(Se necessário) Mapeamento das componentes principais resultantes de volta ao espaço orginal

AVALIAÇÃO DOS CLASSIFICADORES

Avaliação de Classificadores

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Matriz de confusão



COPPE UFRJ

Avaliação de Classificadores

- Acurácia

$$\frac{\text{amostras classificadas corretamente}}{\text{total de amostras}} \rightarrow \frac{VP + VN}{VP + VN + FP + FN}$$

  - Não é confiável para conjunto de dados não-balanceados

- Precisão

$$\frac{\text{amostras positivas classificadas corretamente}}{\text{amostras classificadas como positivas}} \rightarrow \frac{VP}{VP + FP}$$

Avaliação de Classificadores

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Sensibilidade

$$\frac{\text{amostras positivas classificadas corretamente}}{\text{total de amostras positivas}} \rightarrow \frac{VP}{VP + FN}$$

  - Taxa de verdadeiros positivos ou revocação (*recall*)
  - O análogo para a classe negativa é a especificidade

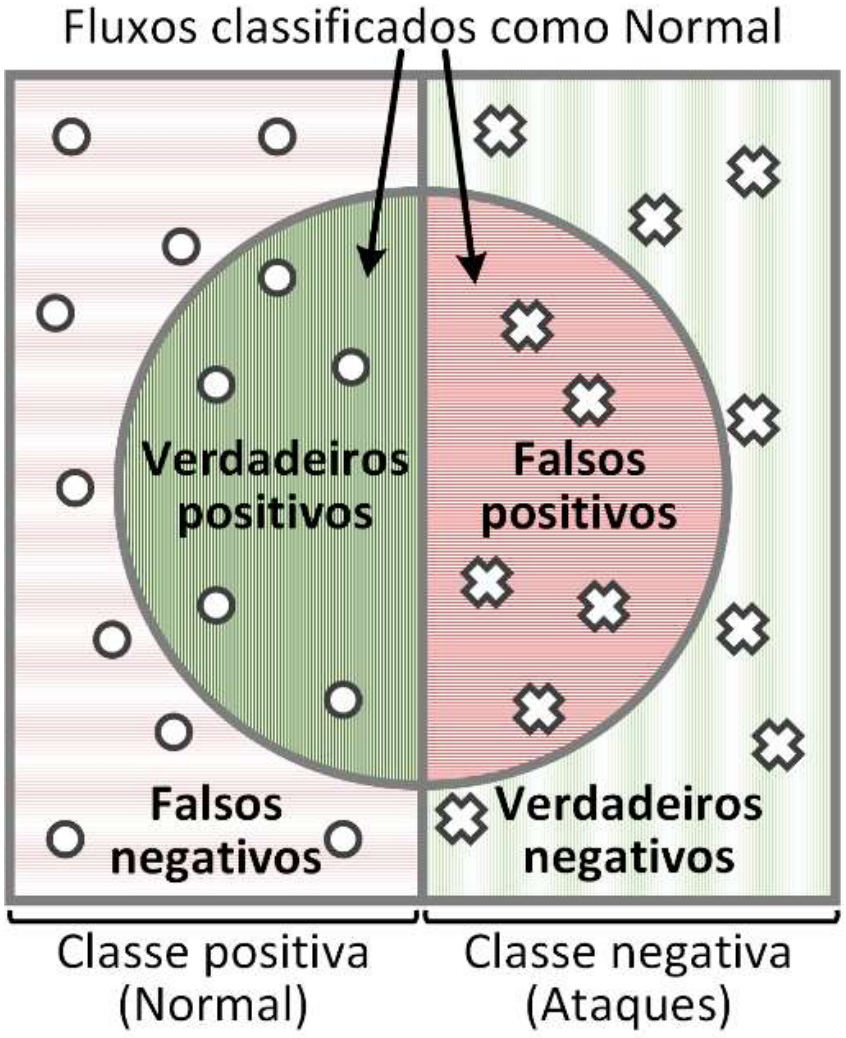
- F1 score
  - Média harmônica da precisão com a sensibilidade

$$\text{F1 score} = \frac{2}{\frac{1}{\text{precisão}} + \frac{1}{\text{sensibilidade}}}$$

COPPE UFRJ

Avaliação de Classificadores

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Precisão =

Sensibilidade =

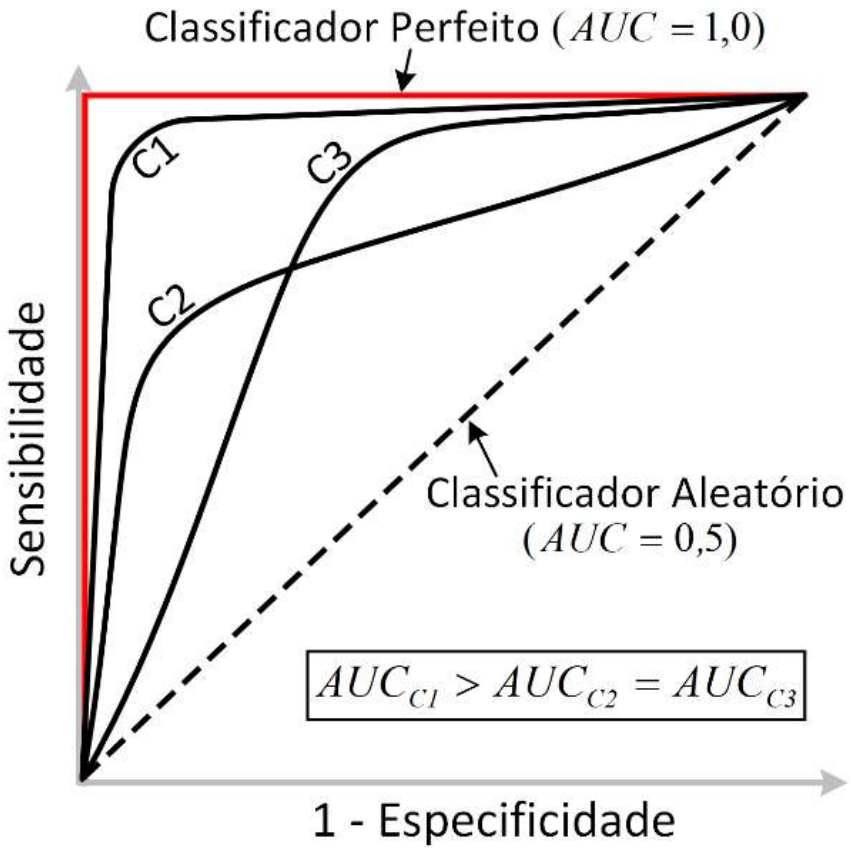
Acurácia =

COPPE UFRJ

Avaliação de Classificadores

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Área abaixo da curva ROC (AUC)
  - Eficaz para dados não-balanceados

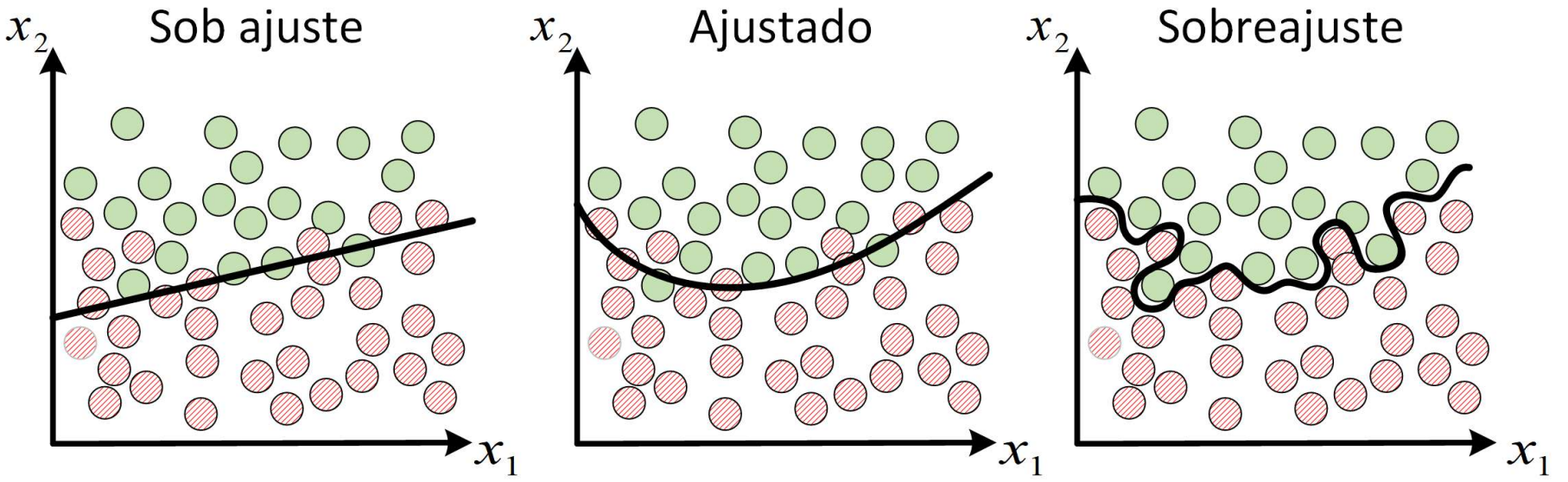


COPPE UFRJ

Métricas de Avaliação de Classificadores

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

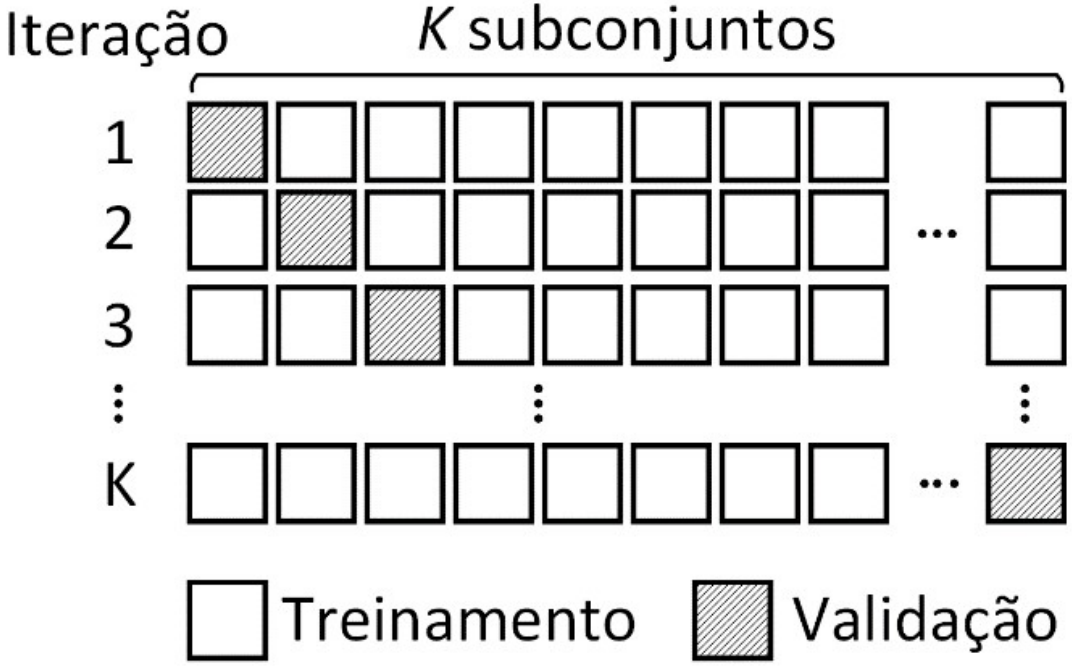
- Objetivo do treinamento é **generalizar bem** o modelo
  - Demasiadamente generalizado: sob ajuste
  - Muito específico para o conjunto de treino: sobreajuste
    - O ruído de um conjunto particular é incorporado ao modelo



COPPE UFRJ

Validação cruzada

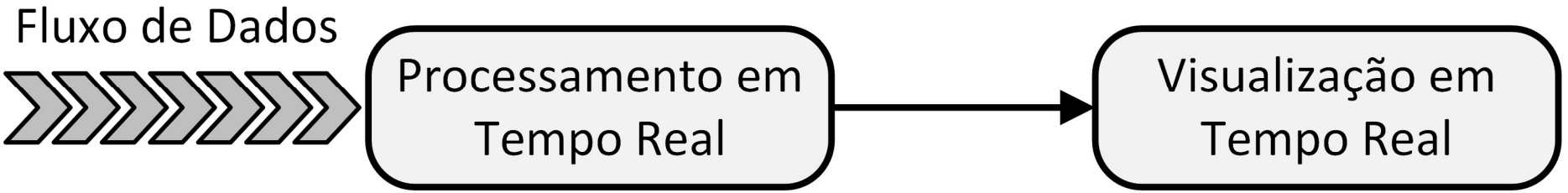
- Eliminação de vícios no treinos, possibilita detectar sobreajuste
  - técnica K-fold é a mais utilizada



ARQUITETURAS PARA PROCESSAMENTO DE FLUXO

Arquitetura Kappa

- Proposta pelo Jay Kreps, co-criador do apache Kafka
- Caminho direto entre
  - Coleta de dados
  - Processamento por fluxos
  - Visualização dos dados em tempo real
- Aplicações com baixa latência que não demandam armazenamento de dados

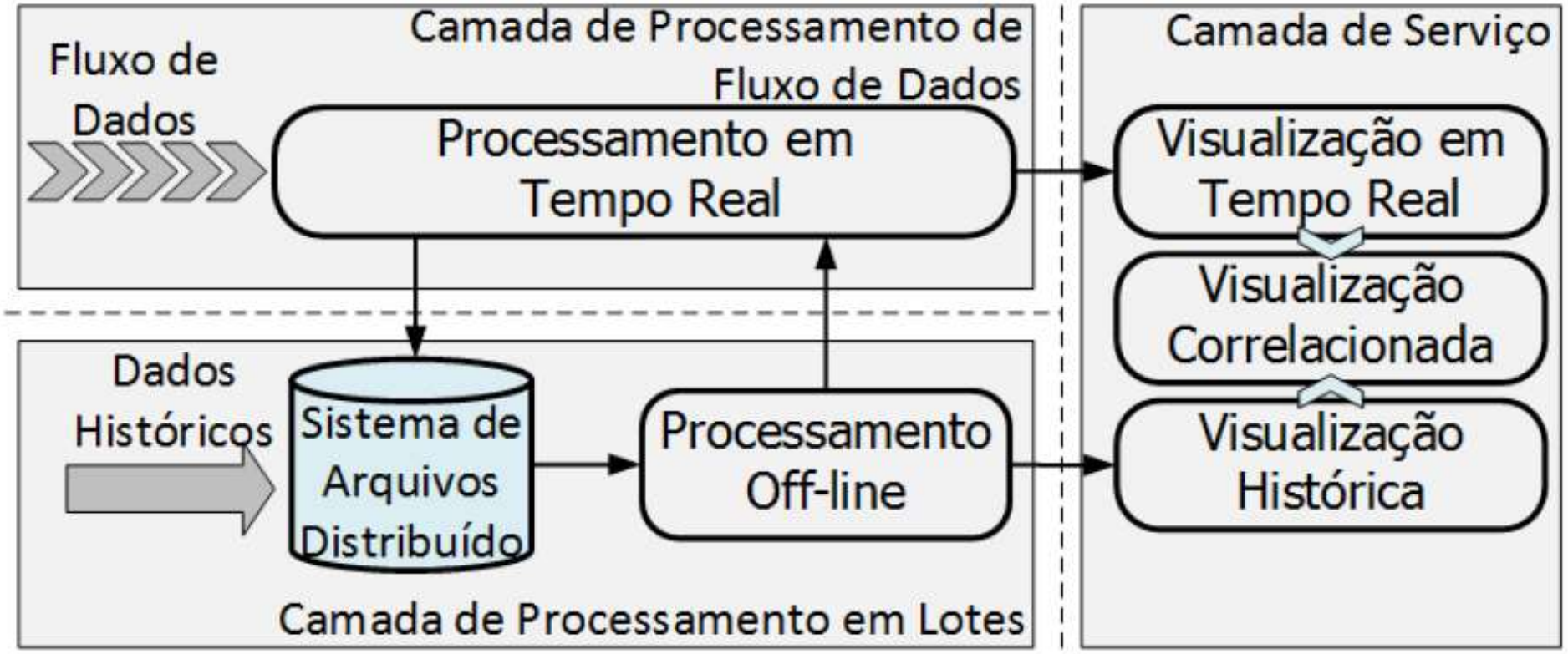


Arquitetura Lambda

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Combina o processamento por fluxo e o processamento em lotes em uma única arquitetura
- **Processamento por fluxos**
  - Resultados aproximados a baixa latência
    - Computação no momento que os dados chegam
    - Contramedidas eficazes no momento da detecção
- **Processamento em lotes**
  - Resultados tardios, porém precisos
    - Classificação *offline*
    - Estimação acurada de parâmetros
    - Atualização *offline* dos parâmetros no modelo online

Arquitetura Lambda



Arquitetura Lambda

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Camada Processamento por Fluxos
  - Núcleo de processamento por fluxos
    - Spark Streaming, Flink, Storm...
  - Registra dados em fluxo em uma base de dados histórica
- Camada Processamento em Lotes
  - Processamento distribuído dos arquivos armazenados
    - MapReduce, Spark, Flink...
  - Base de dados histórica
    - HDFS, Cassandra, Ceph...
- Camada de Serviço
  - Combina a saída as duas camadas anteriores
  - Fornece análise precisa dos resultados históricos e ao mesmo tempo dos resultados em tempo real

COPPE UFRJ

Arquitetura Lambda

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Camada Processamento por Fluxos
  - Núcleo de processamento por fluxos
    - Spark Streaming, Flink, Storm...
  - Registra dados em fluxo em uma base de dados histórica
- Camada Processamento em Lotes
  - Processamento distribuído dos arquivos armazenados

Eficiente para aplicações que usam modelos complexos para processar dados em fluxos

→ **Detecção de Ameaças em Tempo Real**

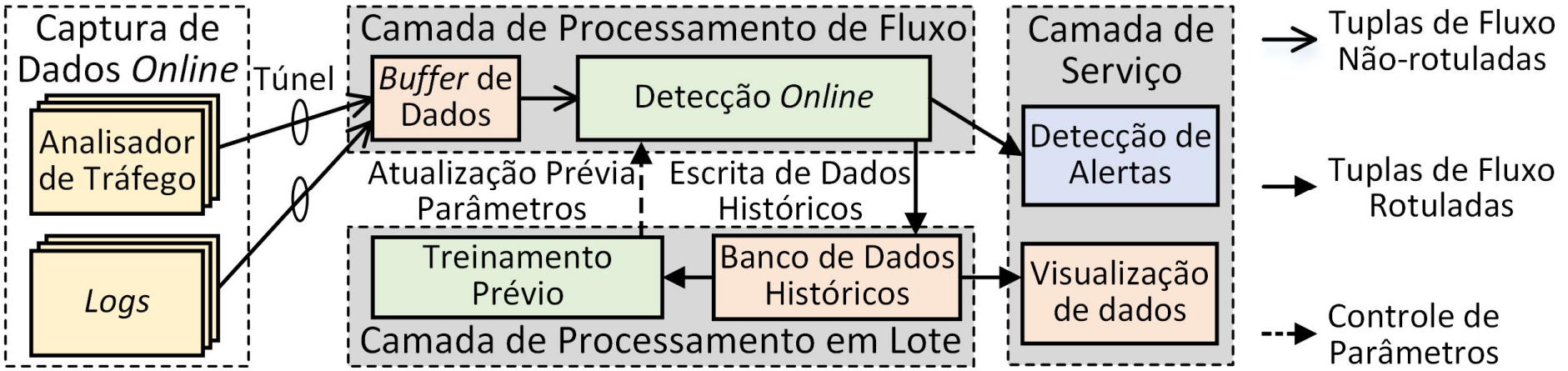
  - Combina a saída as duas camadas anteriores
  - Fornece análise precisa dos resultados históricos e ao mesmo tempo dos resultados em tempo real

COPPE UFRJ

Detecção Online com Treinamento Prévio

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Adaptação da arquitetura lambda para Detecção de Ameaças em Tempo Real



COPPE UFRJ

Algoritmos de Aprendizado de Máquina

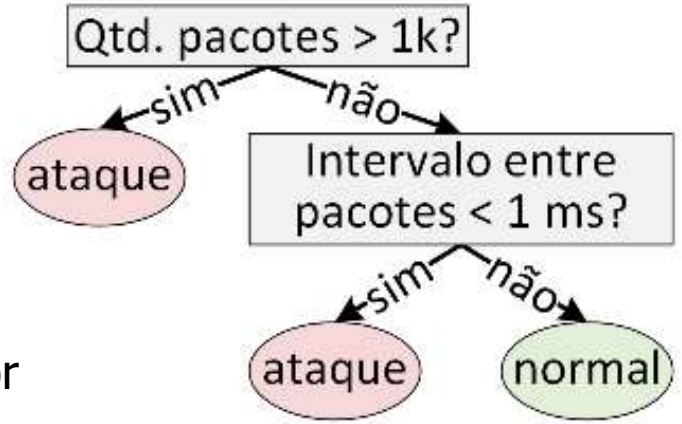
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Árvores de Decisão

Fluxo	Qtd. pacotes	Intervalo médio entre pacotes	Classe
f1	1800	3 ms	ataque
f2	80	2 ms	normal
f3	840	5 ms	normal
f4	230	0,5 ms	ataque

Conjunto de Dados Rotulados

Treinamento do Classificador



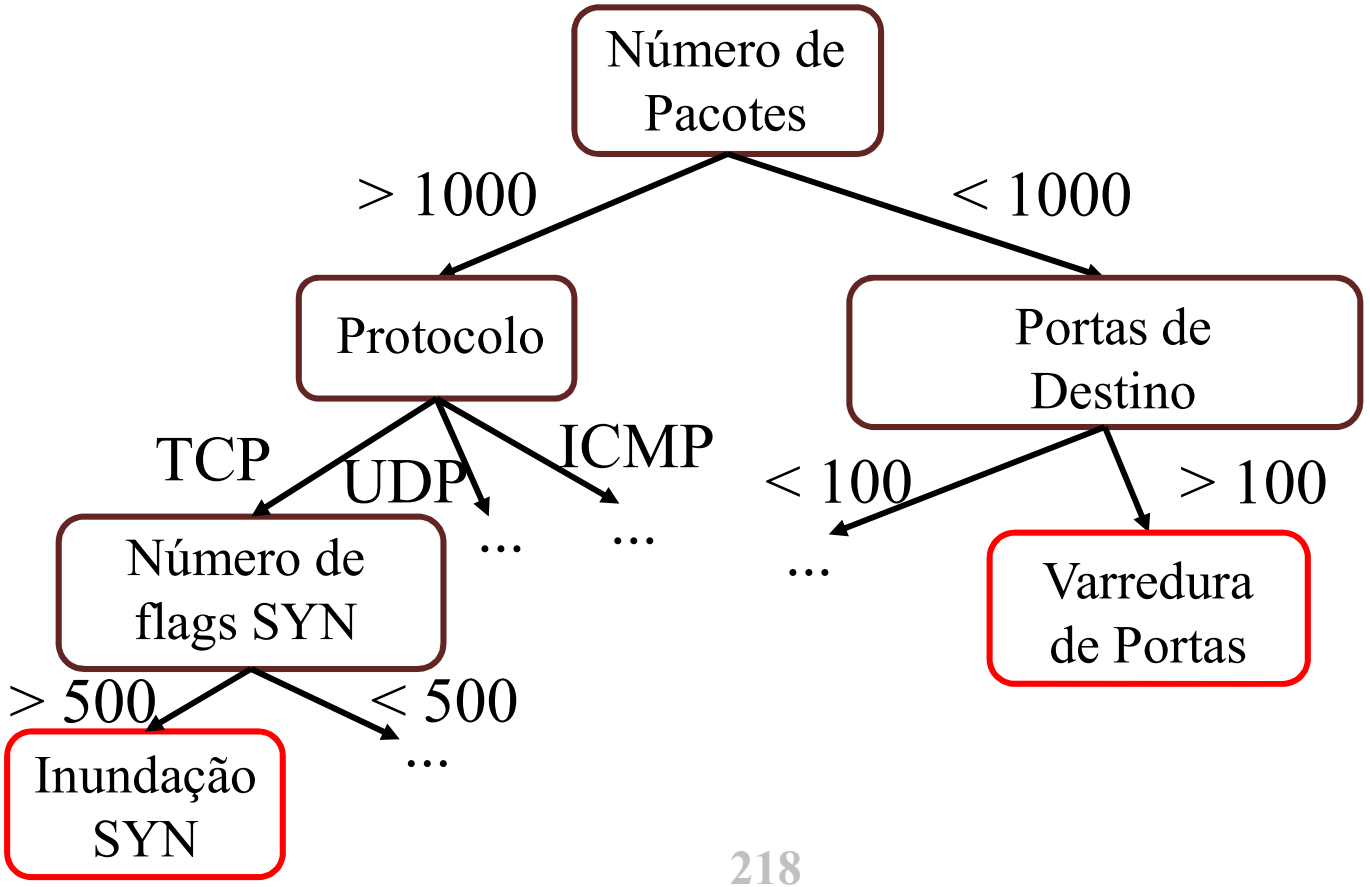
Modelo para Classificação

COPPE UFRJ

Algoritmos de Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Árvores de Decisão

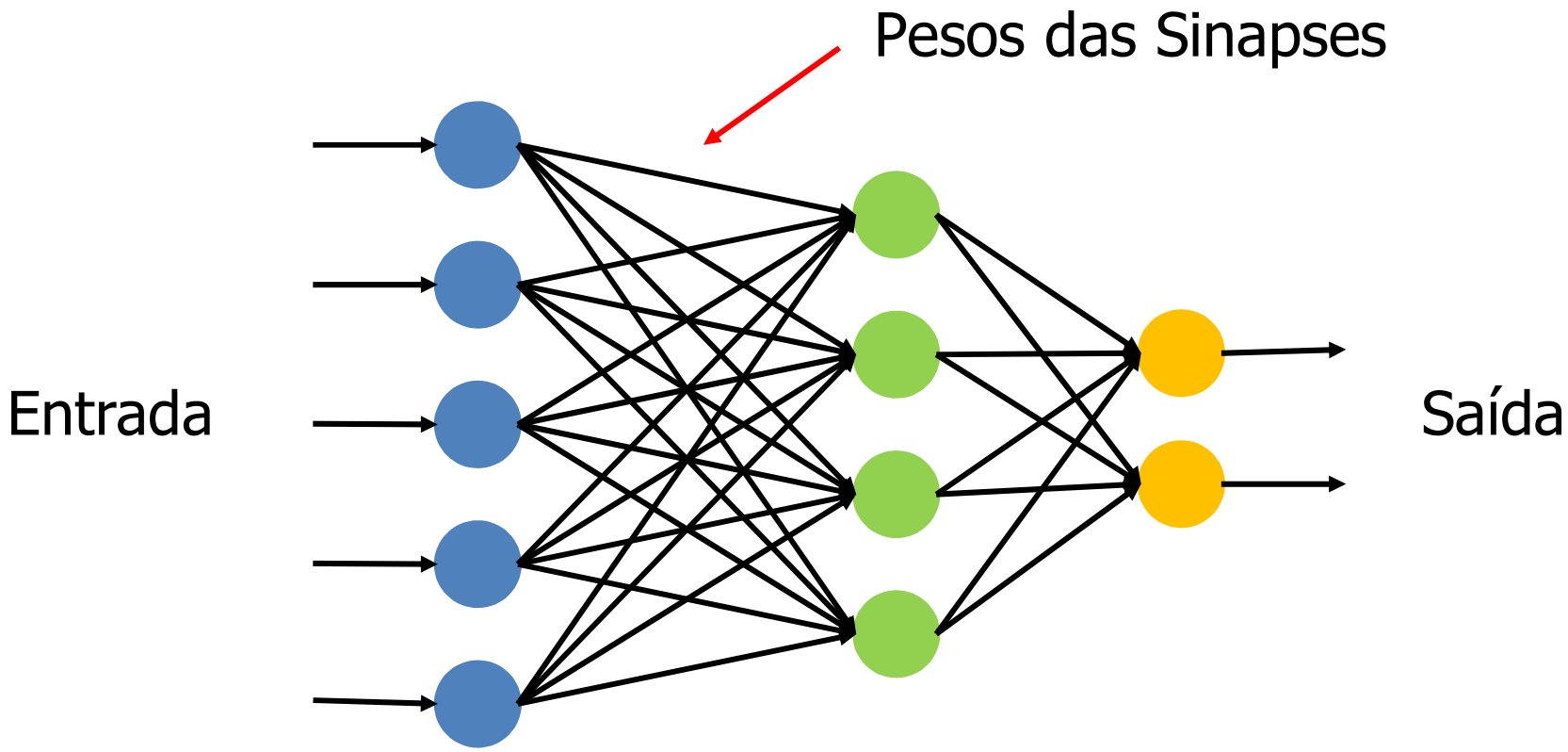


COPPE UFRJ

Algoritmos de Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Redes Neurais

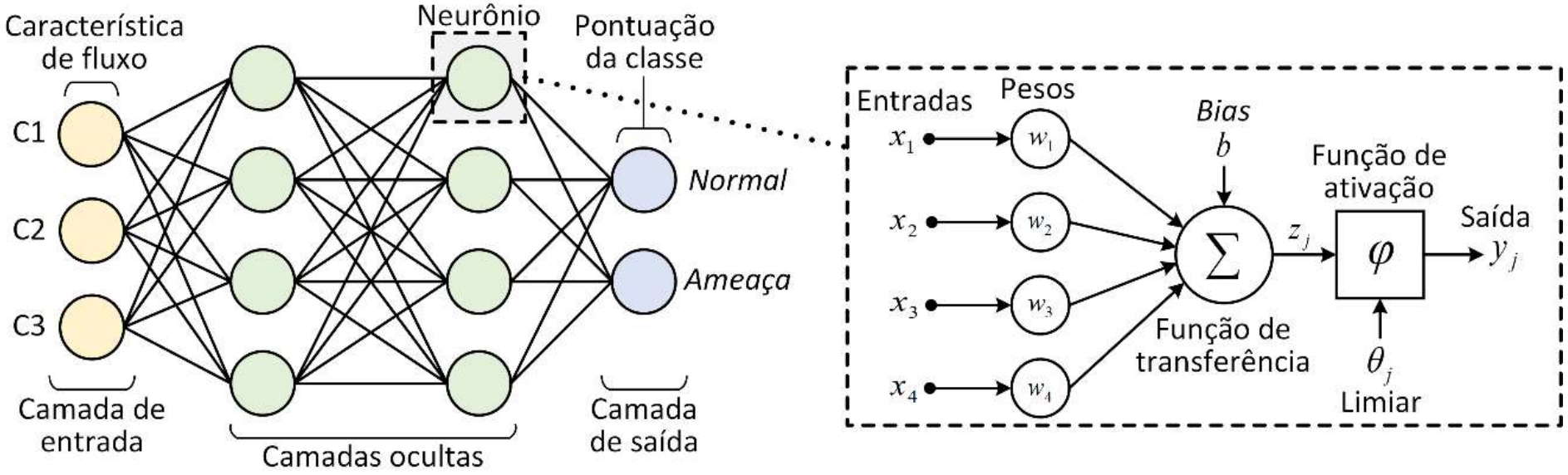


COPPE UFRJ

Algoritmos de Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Redes Neurais

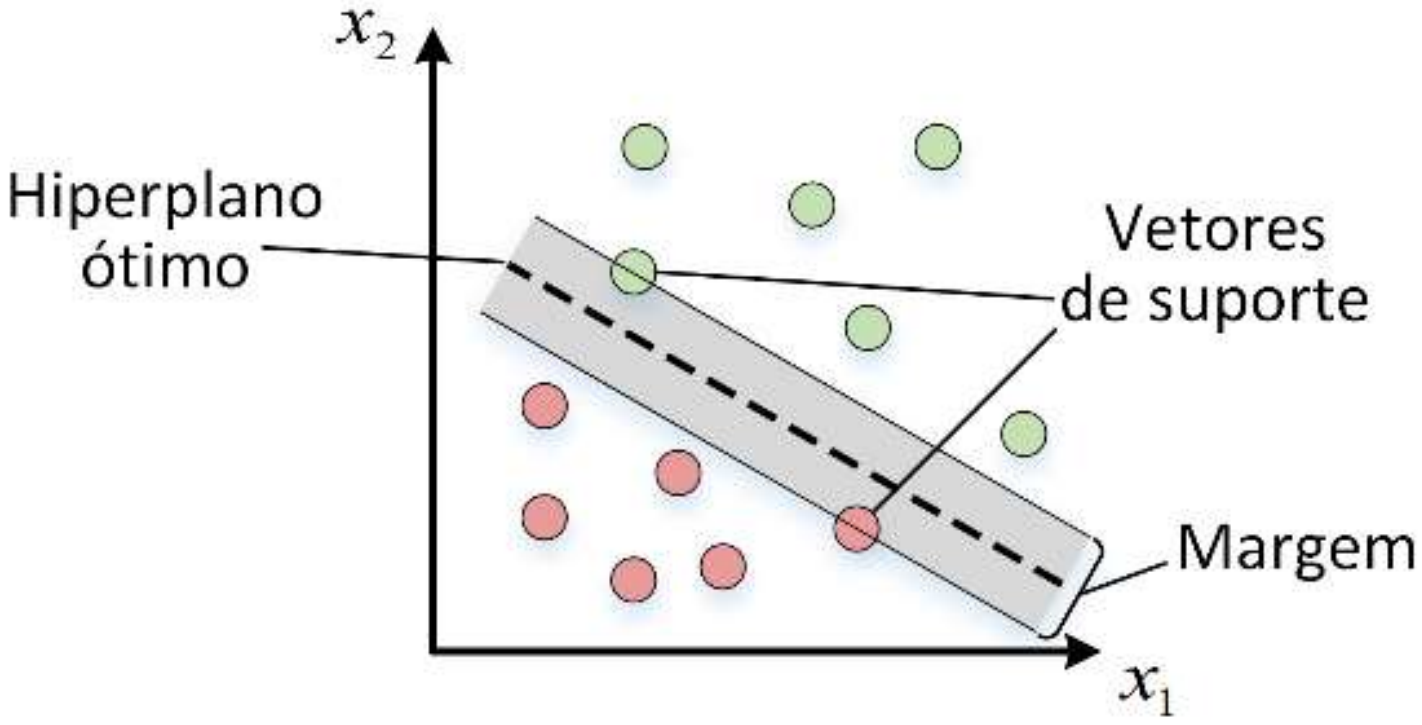


COPPE UFRJ

Algoritmos de Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Máquinas de Vetores de Suporte (SVM)

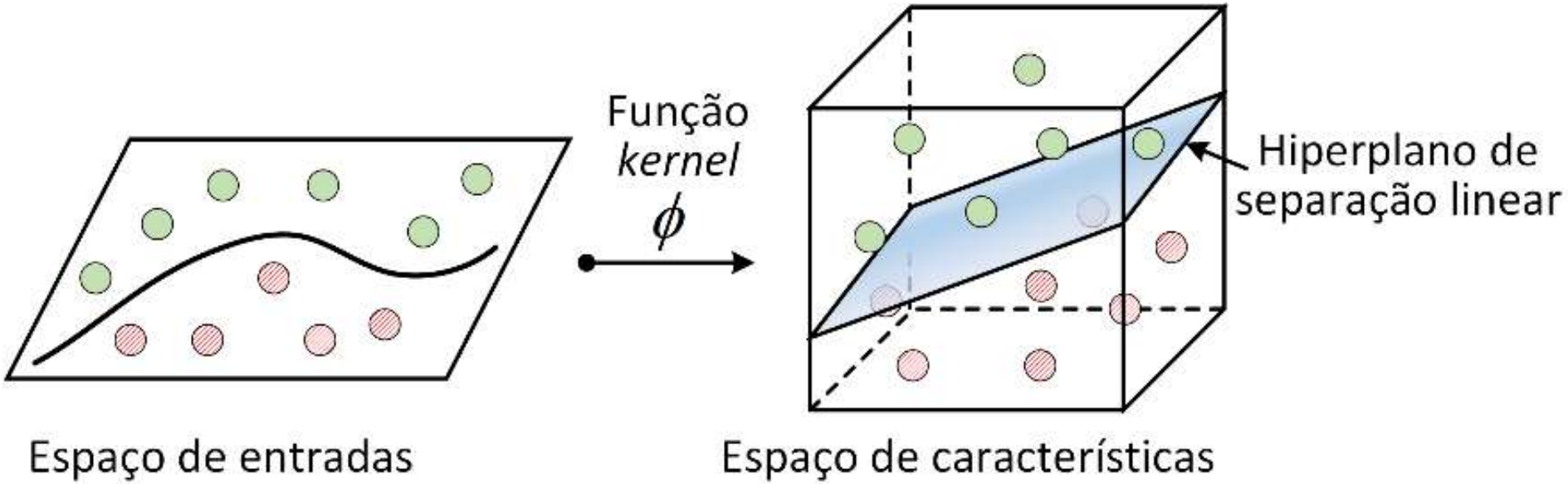


COPPE UFRJ

Algoritmos de Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Artifício de Kernel *(Kernel trick)* para o SVM



COPPE UFRJ

Resultados de Detecção

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Classificação a partir de Treinamento Prévio

Tabela 3.4. Resumo da classificação de ameaças para os algoritmos Árvore de Decisão, Rede Neurais e SVM.

Algoritmo	Conj. Dados	Acurácia	Normal		Ameaça	
			Precisão	Sens.	Precisão	Sens.
Árvore de Decisão	GTA	80.6%	80.7%	94.6%	80.1%	48.8%
	NetOp	92.8%	97.3%	94.8%	51.5%	67.9%
Rede Neural	GTA	96.0%	94.8%	98.8%	97.9%	91.5%
	NetOp	95.1%	97.5%	97.2%	66.7%	69.2%
SVM	GTA	96.3%	95.4%	98.8%	98.0%	92.4%
	NetOp	95.8%	96.5%	98.9%	66.9%	87.2%

COPPE UFRJ

PARTE IV

Detecção de Ameaças Inéditas em Tempo Real a partir de Treinamento Adaptativo

Ataques inéditos e potes de mel

- **Ataques inéditos (*zero-day attack*)** são ataques que acabaram de ser criados e para os quais ainda não existe uma vacina, ou seja, não existe uma assinatura que permita a sua detecção.

- **Potes de mel (honeypot)** é uma ferramenta que simula propositalmente falhas de segurança com o objetivo de colher informações sobre o ataque e o atacante, ou seja, uma espécie de armadilha para invasores.

Ataques inéditos e potes de mel

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Ataques inéditos (*zero-day attack*)** são ataques que acabaram de ser criados e para os quais ainda não existe uma vacina, ou seja, não existe uma assinatura que permita a sua detecção.

- **Potes de mel (honeypot)** é
  que simula propositalmente fa
  com o objetivo de colher infor
  ataque e o atacante, ou seja,
  armadilha para invasores.

COPPE UFRJ

Ataques inéditos e potes de mel

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Ataques inéditos (*zero-day attack*)** são acabaram de ser criados e para os não existe uma vacina, ou seja, não ssinatura que permita a sua

- **Potes de mel (honeypot)** é :a que simula propositalmente fa nça com o objetivo de colher infor ataque e o atacante, ou seja, armadilha para invasores.

COPPE UFRJ

Ataques inéditos e potes de mel

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Proposta

Algoritmos de classificação em linha (o*nline*)

+

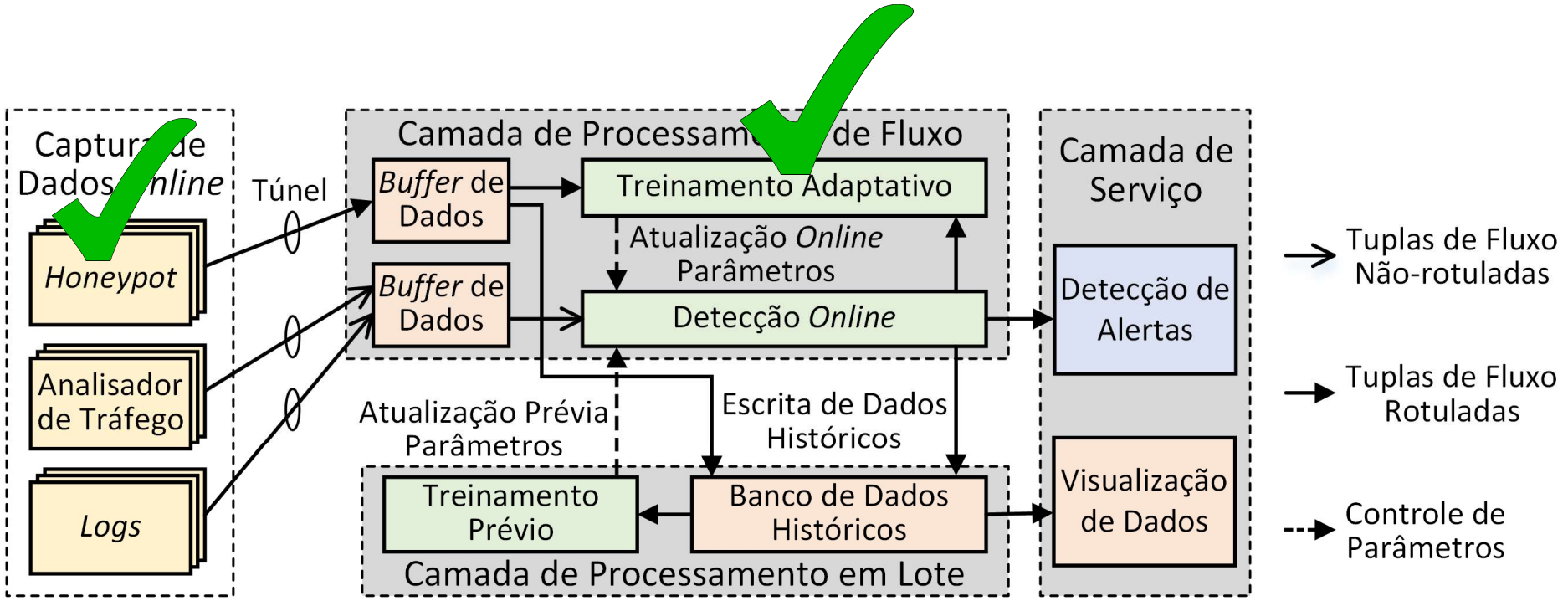
Treinamento adaptativo

=

Detecção de Ameaças Inéditas em Tempo Real

Detecção Online com Treinamento Adaptativo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Algoritmos de classificação em linha (*online*)

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Algoritmos de detecção
  - Algoritmos de classificação em linha (online)
    - Classificação dos ataques com base na captura de comportamento de ataques conhecidos pelo pote de mel (honeypot)
  - Potes de mel (*honeypot*)
    - Monitoramento de atividades maliciosas
      - Hipótese: Somente usuários mal intencionados acessam os potes de mel espalhados na Internet
        - Não possuem serviços válidos

COPPE UFRJ

Métodos de detecção de ameaças adaptativas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Classificação Online
  - Dados marcados de acordo com sua fonte
  - Dados dos Honeypot → Sempre ameaças
    - Analisador de tráfego → Classificado pelos algoritmos

Métodos de detecção de ameaças adaptativas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Classificação Online
  - Dados marcados de acordo com sua fonte
  - Dados dos Honeypot → Sempre ameaças
    - Analisador de tráfego → Classificado pelos algoritmos
  - Dados dos Honeypot atualizam os parâmetros dos algoritmos
    - Aprendem novas ameaças e mudanças de comportamento do invasor
  - Dados do analisador de Trafego
    - Classificado como ameaça
      - Alerta
    - Classificado como normal
      - Atualização dos parâmetros do algoritmo

Classificação Online

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Gradiente descendente estocástico com Momentum
  - Regressão Logística
    - Função Sigmoid → Valores entre 0 e 1
  - Função de custo convexo
    - Em cada amostra, um passo em direção à função de custo mínimo

Classificação Online

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Gradiente descendente estocástico com Momentum
  - Regressão Logística
    - Função Sigmoid → Valores entre 0 e 1
  - Função de custo convexo
    - Em cada amostra, um passo em direção à função de custo mínimo

- Máquina de vetores de suporte on-line
  - Aproximação de margem suave para os limites de decisão
    - Função de perda de articulação (hinge-loss)→ também convexa

Máquina de Vetores de Suporte Incremental

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Máquina de Vetores de Suporte Incremental (*online*)
  - Aproximação de variáveis de folga (*soft margin*) para decisões de fronteira
    - Função perda de articulação (*hinge-loss*) ➔ convexa

Máquina de Vetores de Suporte Incremental

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

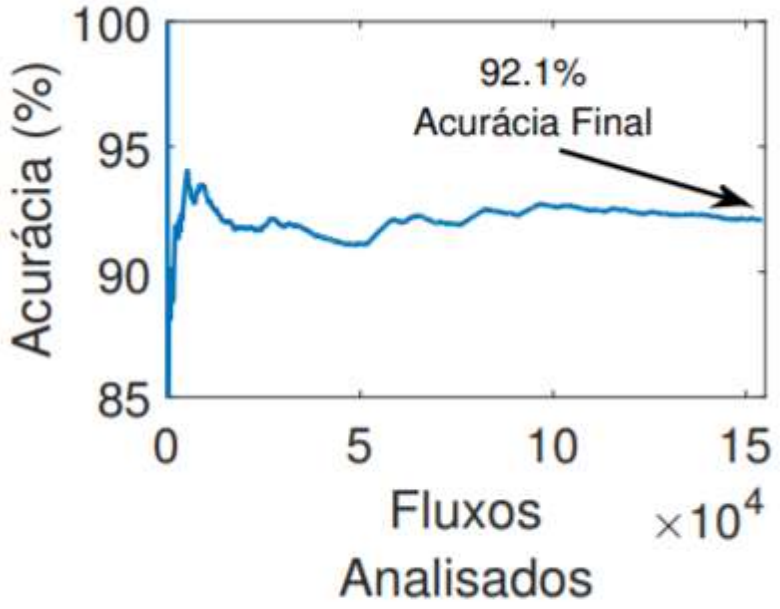
Algoritmo 2: Máquina de Vetores de Suporte Incremental.

Entrada: Características de fluxo de entrada x, Classe y
Saída : Classe prevista *predict*, parâmetros do treinamento θ

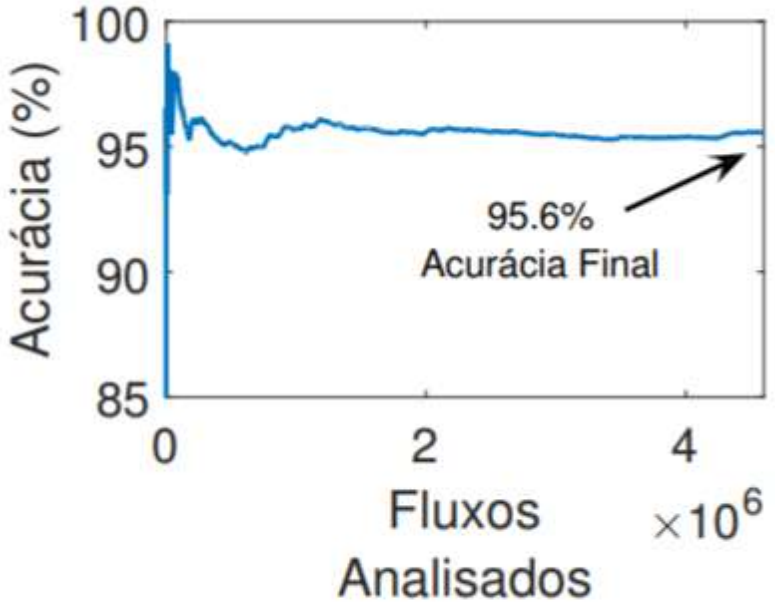
Inicializar θ, α, λ;
for $i \leftarrow 1$ **to** m **do**
  $predict = sign(\theta^T x_{(i)})$;
  if $predict == 1$ *and* $y_{(i)} == -1$ **then**
    Envia Alerta;
  else
    if $y_{(i)}\theta^T x_{(i)} > 1$ **then**
      $\nabla_{(i)} = \theta$;
    else
      $\nabla_{(i)} = -\lambda y_{(i)} x_{(i)}$;
    end
    $\theta = \theta - \alpha\nabla_{(i)}$
  end
end

Máquina de Vetores de Suporte Incremental

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



(a) Conjunto de dados do Laboratório.



(b) Conjunto de dados do Operador.

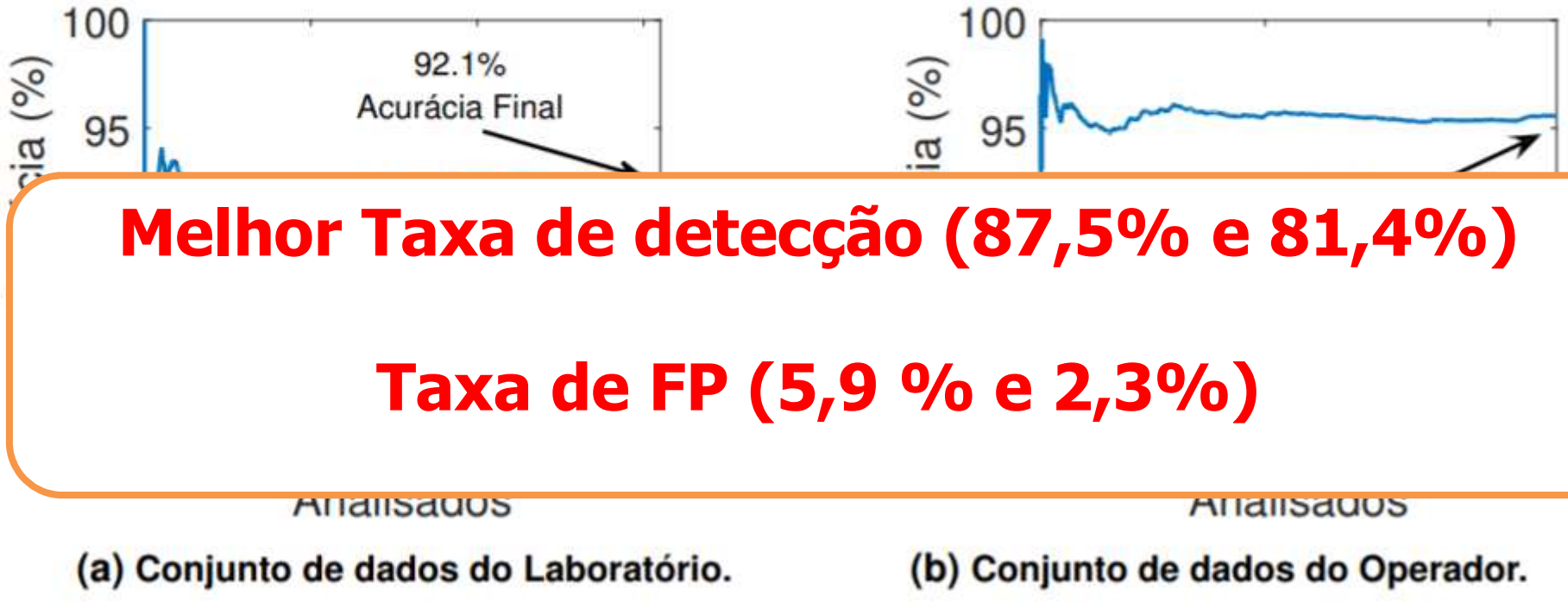
	Normal	Ameaça
Normal	100658	6297
Ameaça	5900	41332

	Normal	Ameaça
Normal	4099766	94654
Ameaça	117051	512112

COPPE UFRJ

Máquina de Vetores de Suporte Incremental

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Melhor Taxa de detecção (87,5% e 81,4%)

Taxa de FP (5,9 % e 2,3%)

(a) Conjunto de dados do Laboratório.

	Normal	Ameaça
Normal	100658	6297
Ameaça	5900	41332

(b) Conjunto de dados do Operador.

	Normal	Ameaça
Normal	4099766	94654
Ameaça	117051	512112

COPPE UFRJ

Gradiente Descendente Estocástico com Momento

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- O método do Gradiente Descendente (GD) é um algoritmo de otimização que baseia-se na propriedade do gradiente para indicar a direção de máximo crescimento da função
- Gradiente Descendente com Momento (GDM) acrescenta um valor no ajuste do erro para acelerar a convergência
- Gradiente Descendente Estocástico com Momento é uma aproximação do GDM que requer apenas uma amostra para atualizar um parâmetro em uma iteração particular
  - Regressão logística
    - Função Sigmoid → Valores entre 0 e 1
  - Função de custo convexa
    - A cada amostra um degrau na direção do mínimo

Gradiente Descendente Estocástico com Momento

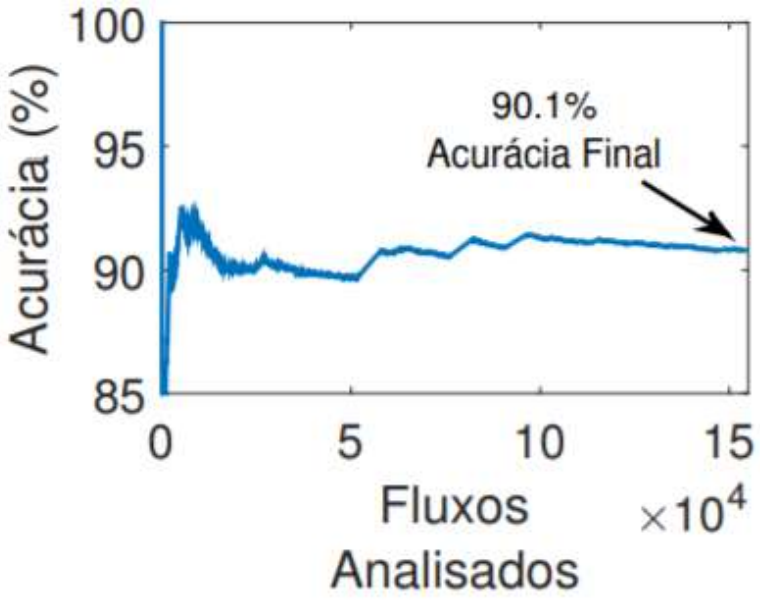
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Algoritmo 1: Gradiente Estocástico Descendente com Momento.

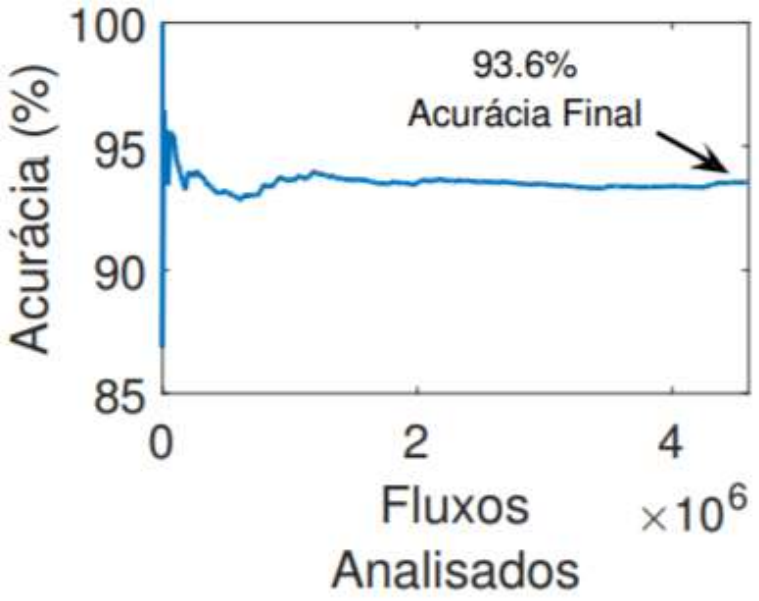
Entrada: Características de fluxo de entrada x, Classe y
Saida : Classe Prevista *predict*, Parâmetros do treinamento θ

Inicializar θ, $\Delta\theta$, α, β;
for $i \leftarrow 1$ **to** m **do**
  $h_\theta(x_{(i)}) = \frac{1}{1+e^{-\theta^T x_{(i)}}}$;
  $predict = round(h_\theta(x_{(i)}))$;
  if $predict == 1$ *and* $y_{(i)} == 0$ **then**
    Envia Alerta;
  else
    $\theta = \theta - \alpha\nabla J_{(i)}(\theta) + \beta\Delta\theta$;
    $\Delta\theta = \alpha\nabla J_{(i)}(\theta)$;
  end
end

Gradiente Descendente Estocástico com Momento

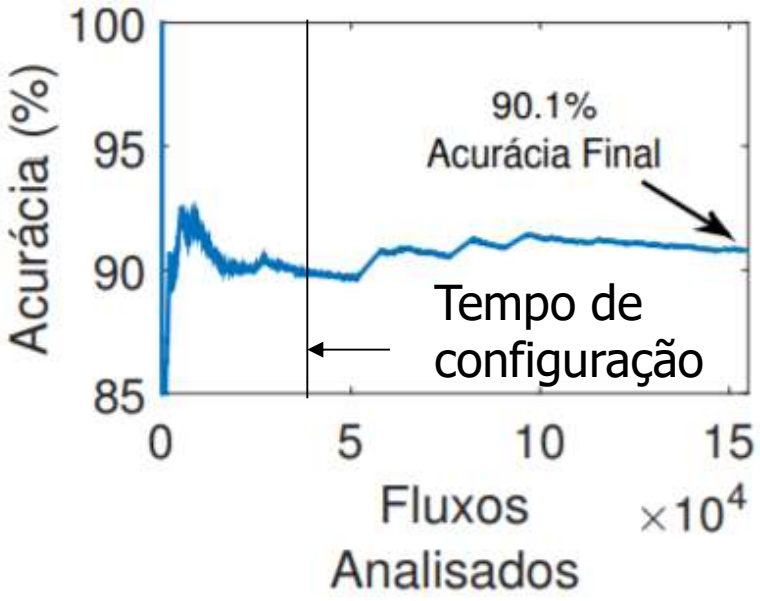


(a) Conjunto de dados do Laboratório.

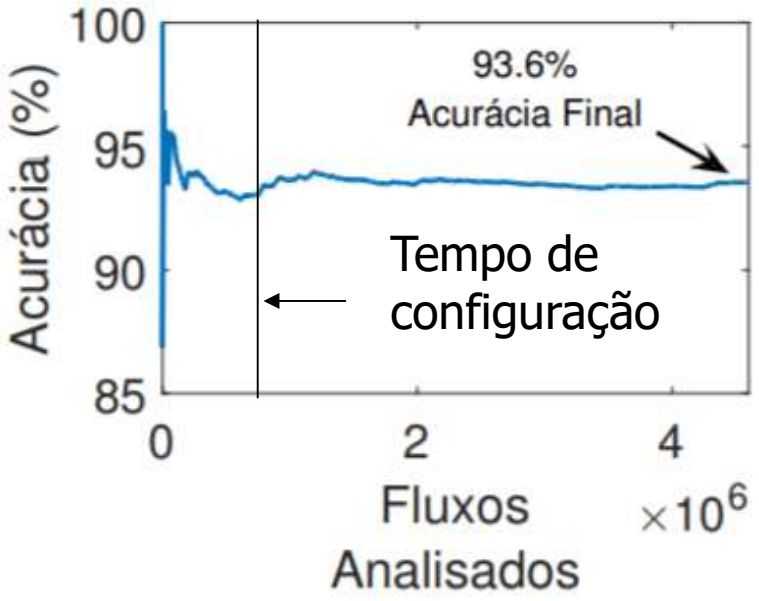


(b) Conjunto de dados do Operador.

Gradiente Descendente Estocástico com Momento

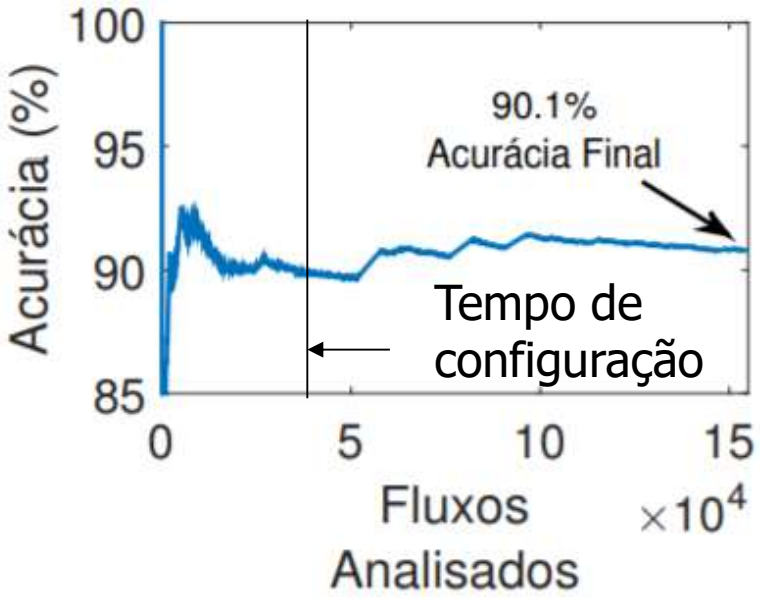


(a) Conjunto de dados do Laboratório.

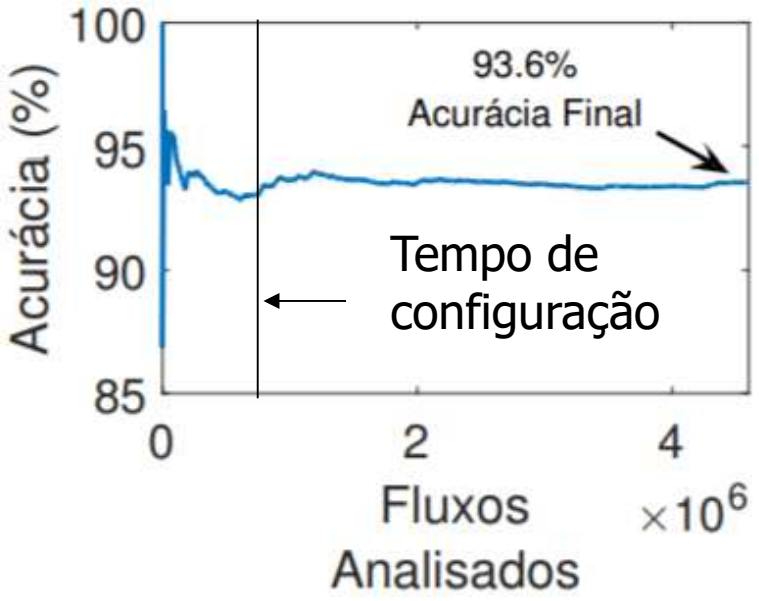


(b) Conjunto de dados do Operador.

Gradiente Descendente Estocástico com Momento



(a) Conjunto de dados do Laboratório.



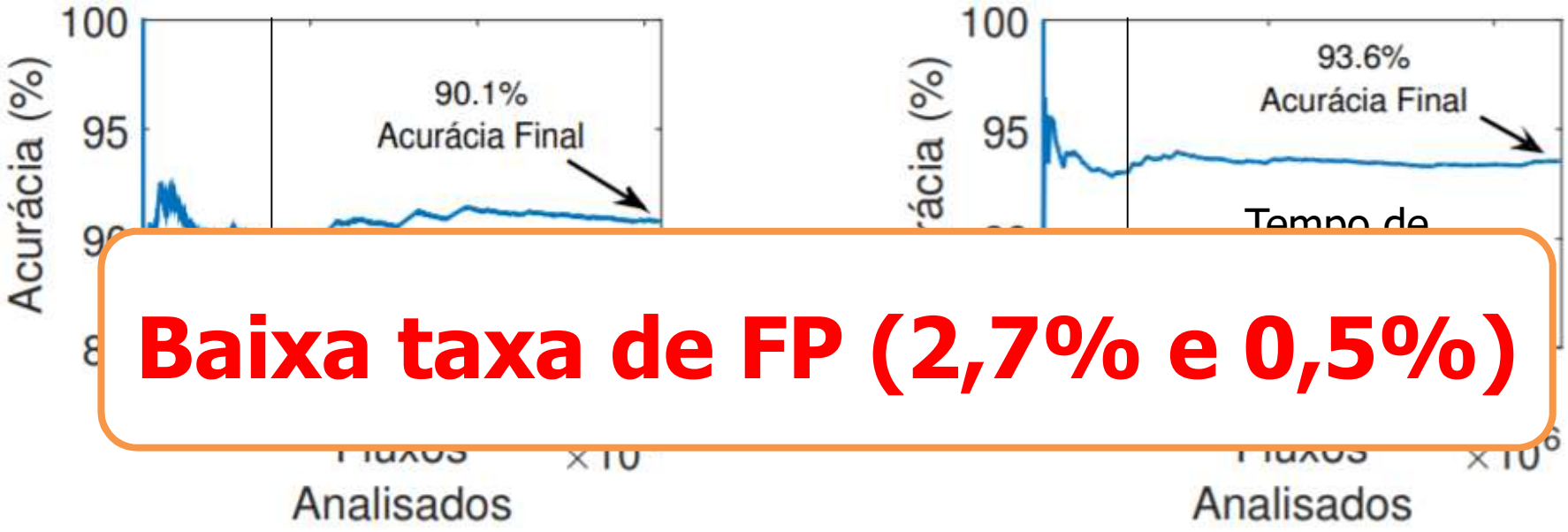
(b) Conjunto de dados do Operador.

	Normal	Ameaça
Normal	104028	2927
Ameaça	11245	35987

	Normal	Ameaça
Normal	4172989	21431
Ameaça	287441	341722

Gradiente Descendente Estocástico com Momento

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Baixa taxa de FP (2,7% e 0,5%)

(a) Conjunto de dados do Laboratório.

(b) Conjunto de dados do Operador.

	Normal	Ameaça
Normal	104028	2927
Ameaça	11245	35987

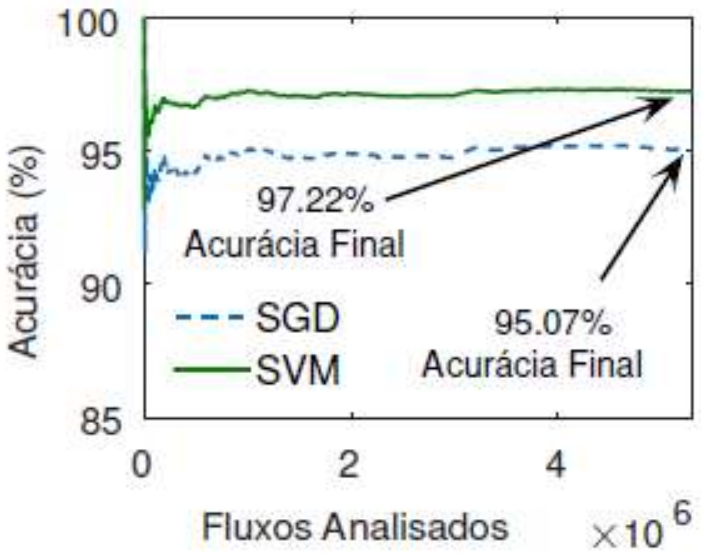
	Normal	Ameaça
Normal	4172989	21431
Ameaça	287441	341722

COPPE UFRJ

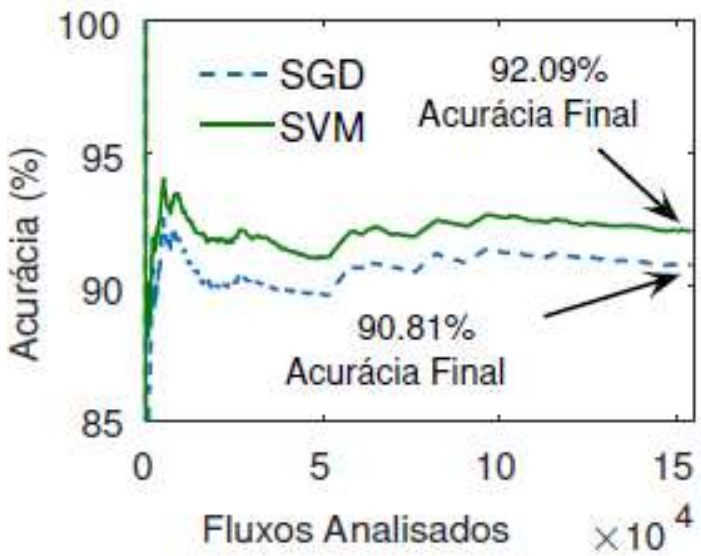
Resultados de Detecção com Treinamento Adaptativo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Stochastic Gradient Descent (SGD) vs. Máquina de Vetores de Suporte Incremental(ISVM)



Conjunto de dados GTA/UFRJ



Conjunto de dados NetOp

COPPE UFRJ

Esquemas adaptativos de detecção de ameaças

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Detecção por anomalia**
  - Detecção de ameaça baseada em comportamento de usuário legítimo
    - Ameaças apresentam um comportamento diferente dos usuários legítimos
    - Comportamento legítimo atualizado em tempo real
  - Protege a rede contra ameaças inéditas (*zero-day threats*)
    - Ameaças inéditas previamente detectadas no pote de mel

COPPE UFRJ

Detecção de Anomalia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- **Distribuição Normal**
  - Calcula parâmetros Gaussianos para cada característica
    - Assume que é ameaça quando as parâmetros das amostras diferem acima de um determinado patamar da distribuição Normal
- **Série temporal de entropia**
  - Medida de dispersão em um valor
  - Entropia calculada para uma janela deslizante de fluxo
    - Assume que é uma ameaça quando a entropia desvia do comportamento normal da entropia

Detecção de Anomalia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Detecção de ameaças inéditas (*zero-day attacks*)
  - Estimativa de comportamento padrão
  - Alerta para amostras fora desse comportamento

- Detecção de Anomalias pela análise do valor de entropia
  - Conjunto de dados dividido em dois:
    - Tráfego normal
      - 70% para treino
      - 30% para determinar falsos positivos
    - Tráfego de ataque para determinar taxa de detecção

Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Hipótese → Caso Trivial
  - Distribuição das características dos fluxos na rede é aproximada por uma gaussiana
- Distância das características de uma amostra para a média de uma distribuição normal
- Condições para detecção anomalia

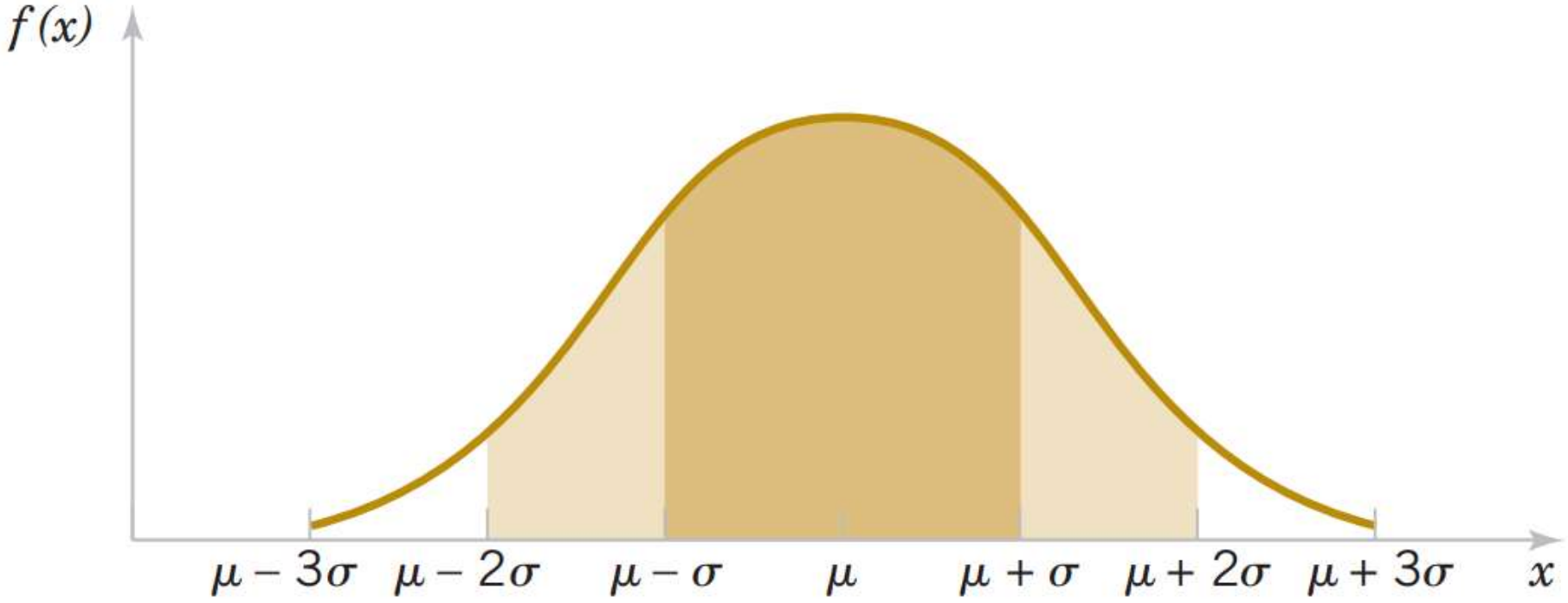
$$X_j > \mu_j + limiar * \sigma_j^2 \quad \text{ou} \quad X_j < \mu_j - limiar * \sigma_j^2$$

  - *limiar* → Número de desvios padrão que se permite a distanciação da amostra à média da normal

Anomalia = amostra que ao menos uma característica atende a condição para detecção

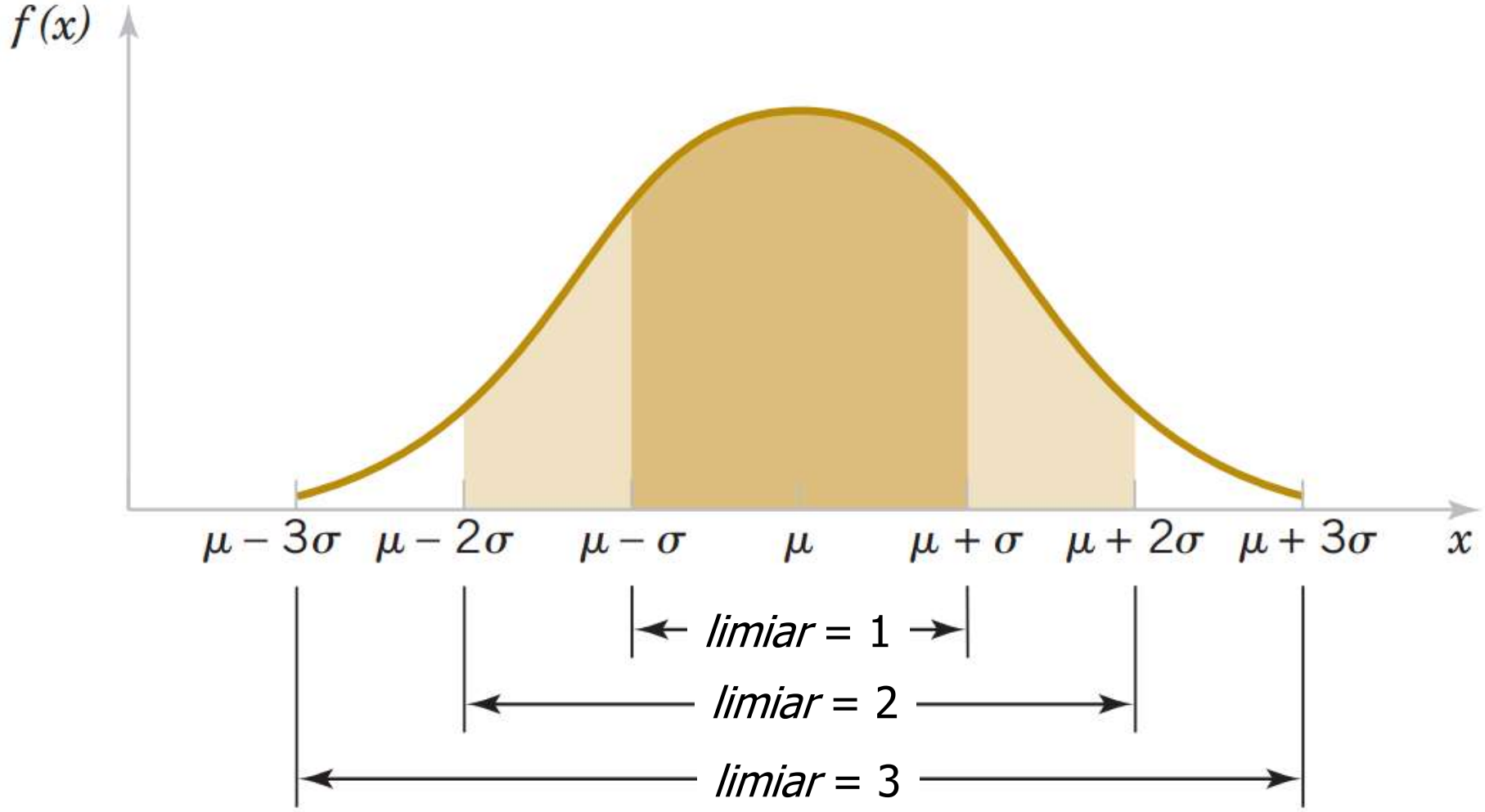
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



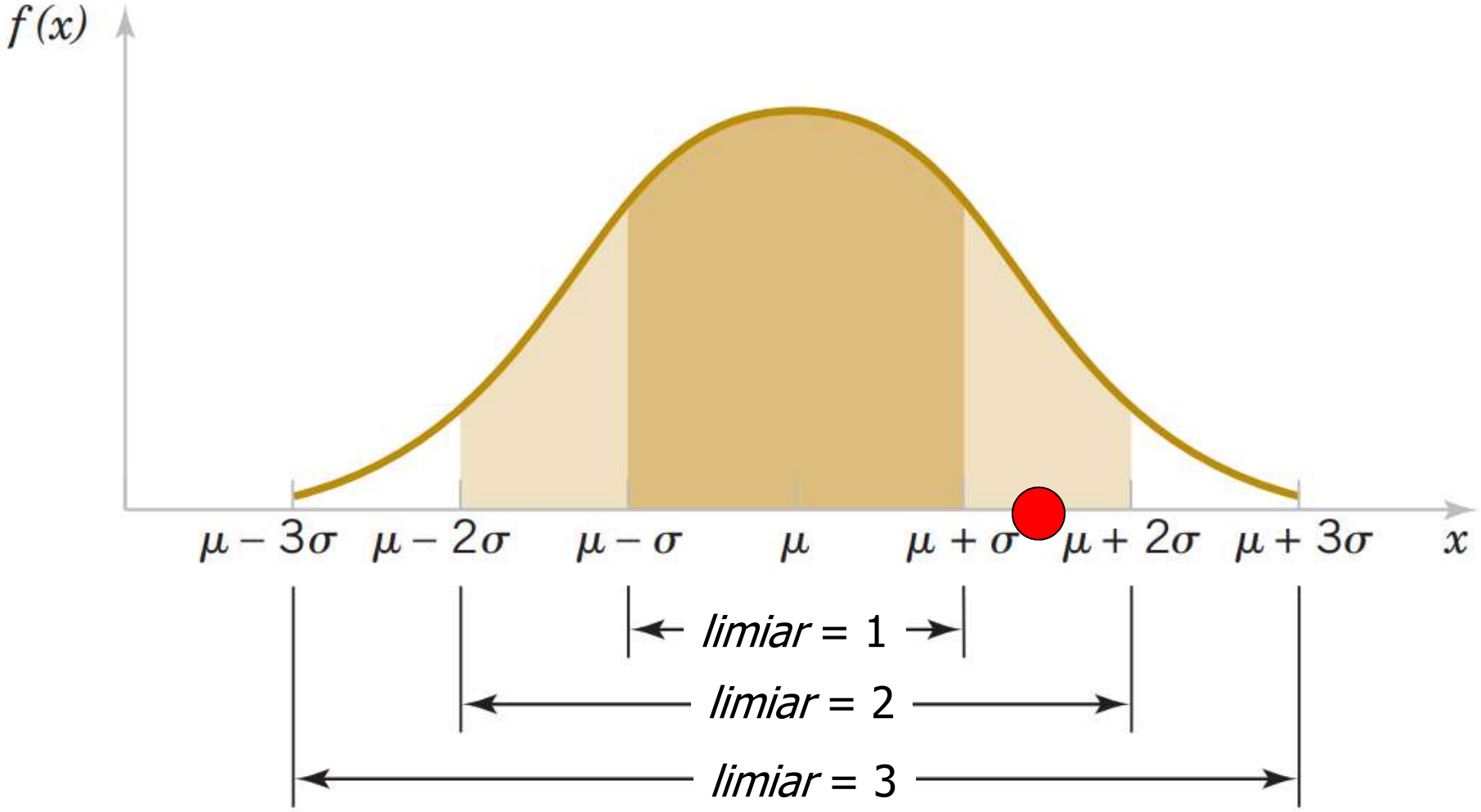
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



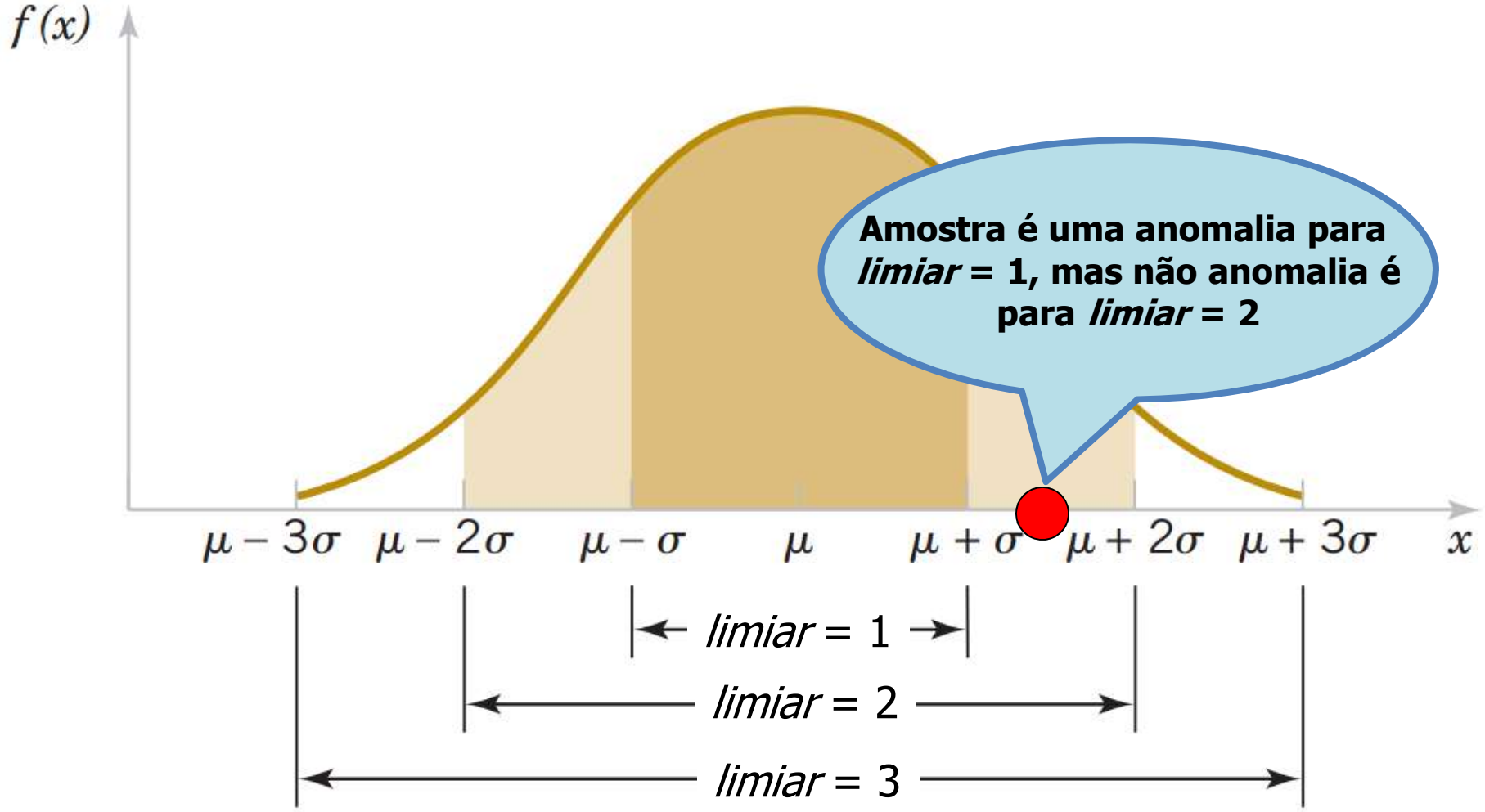
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



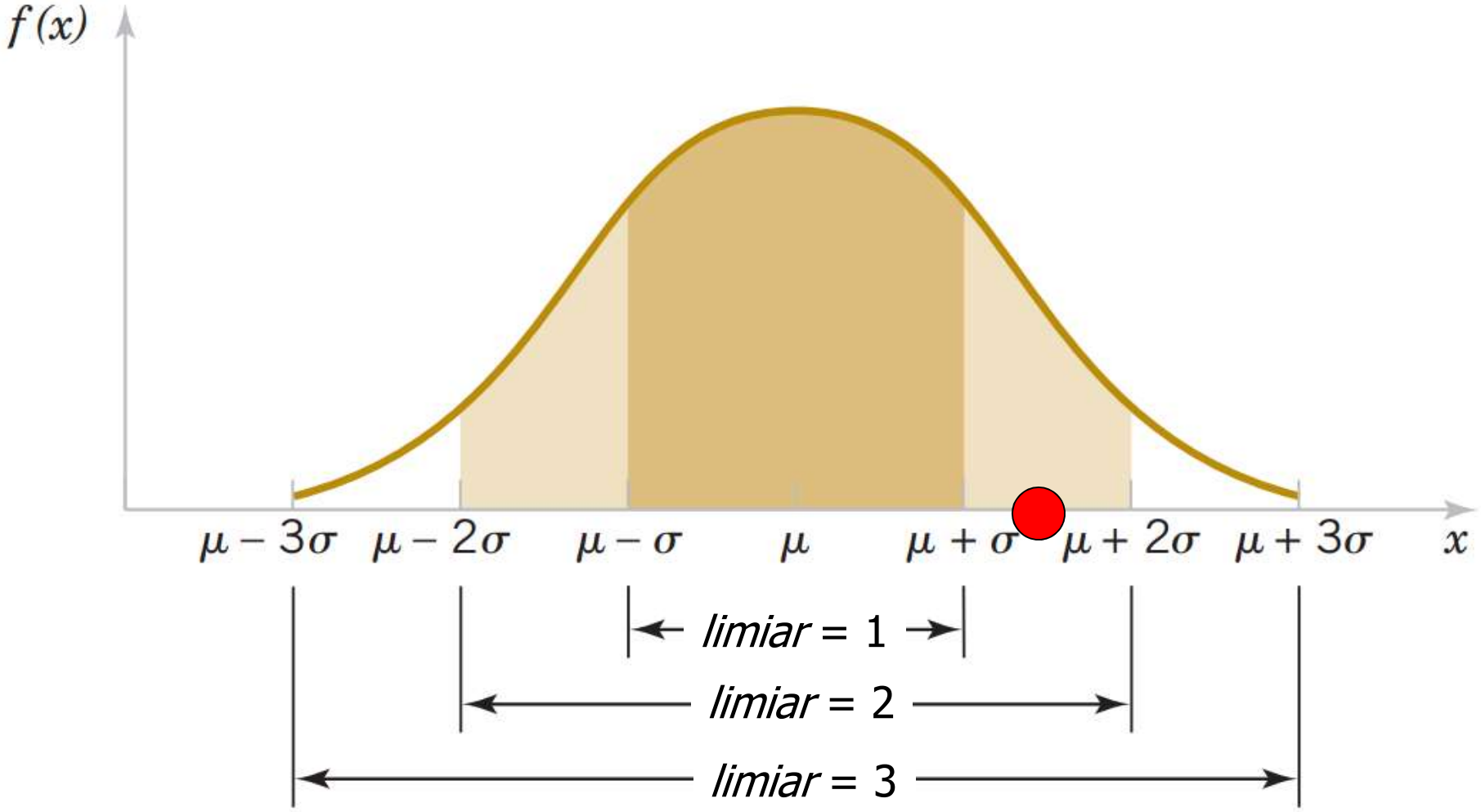
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



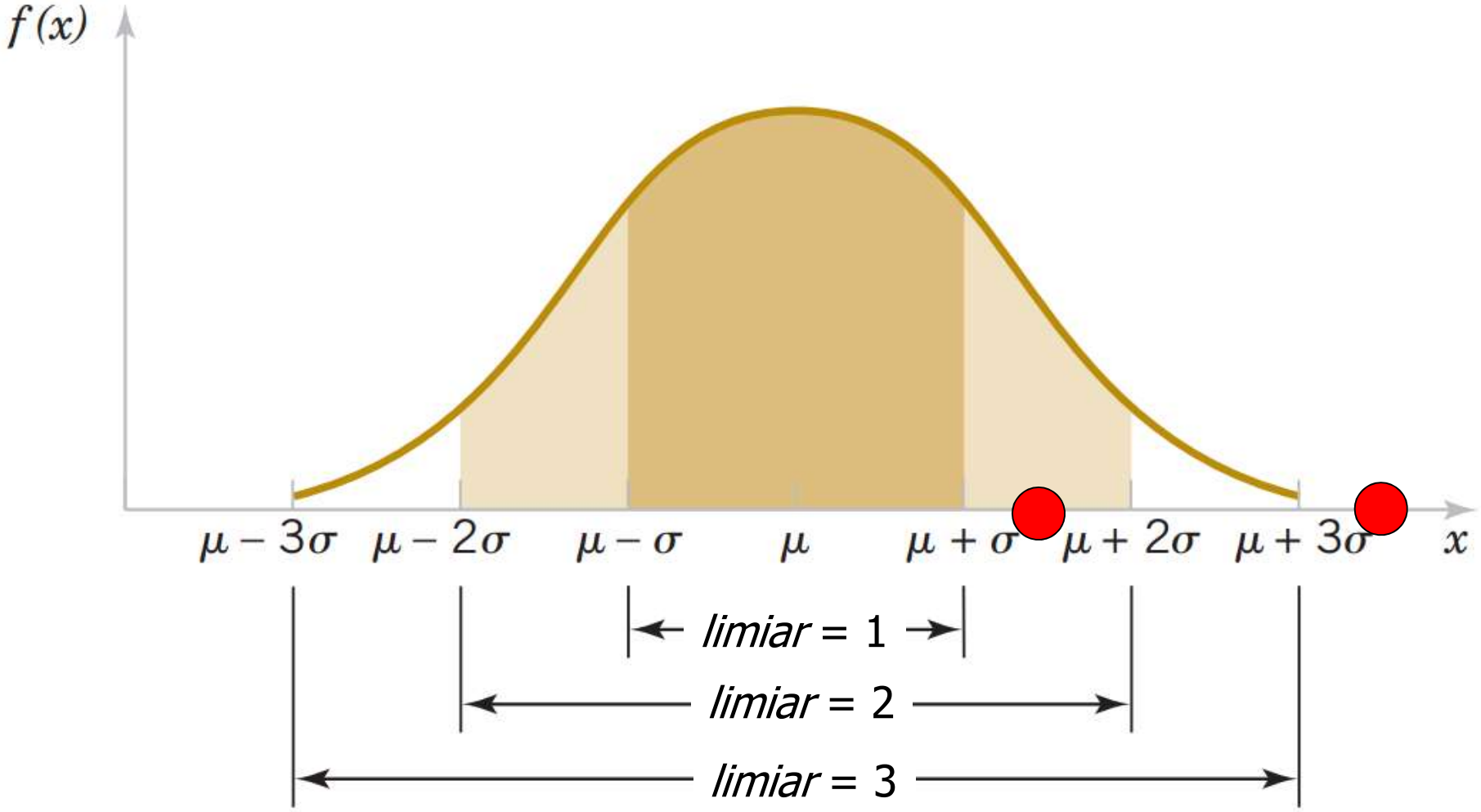
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



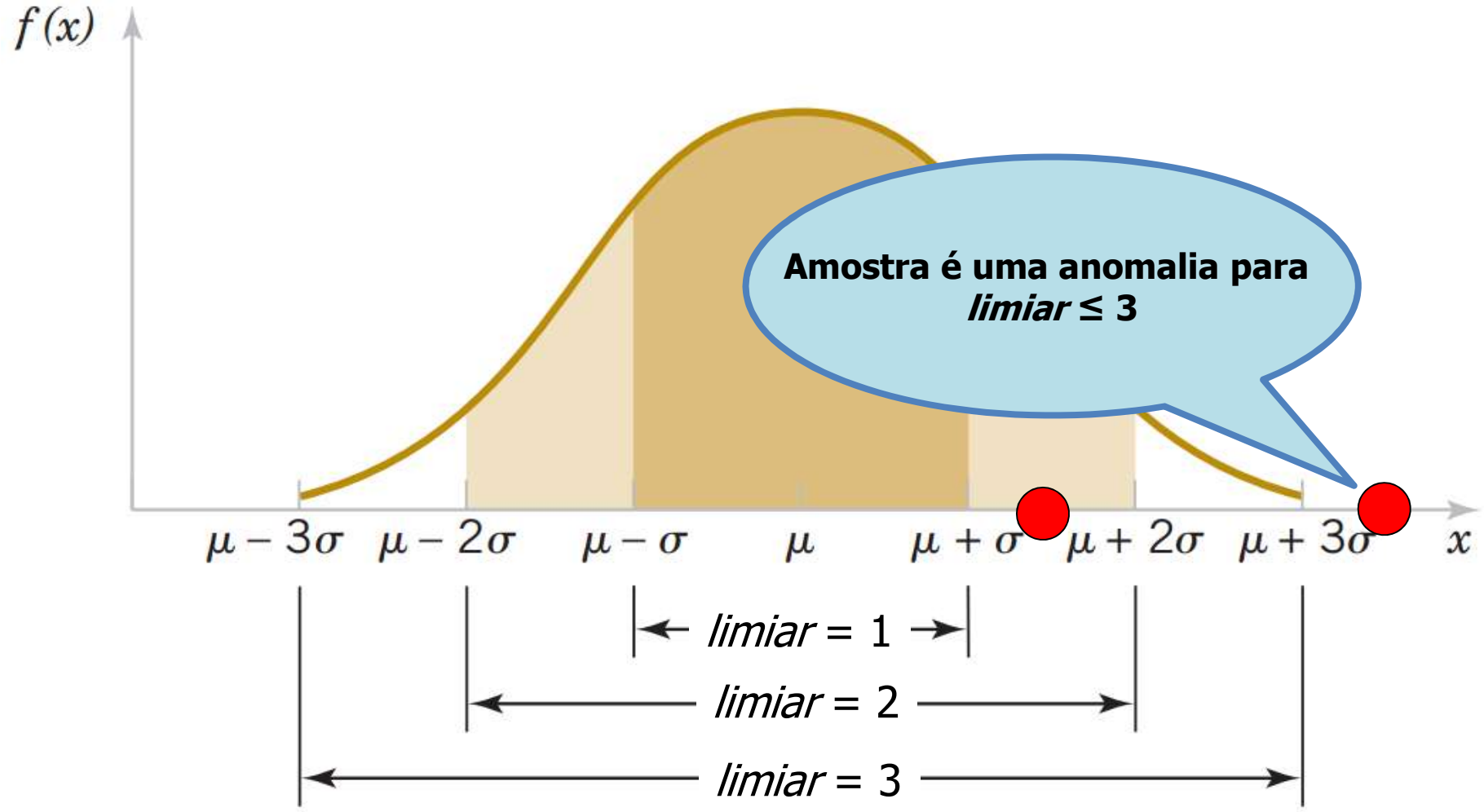
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



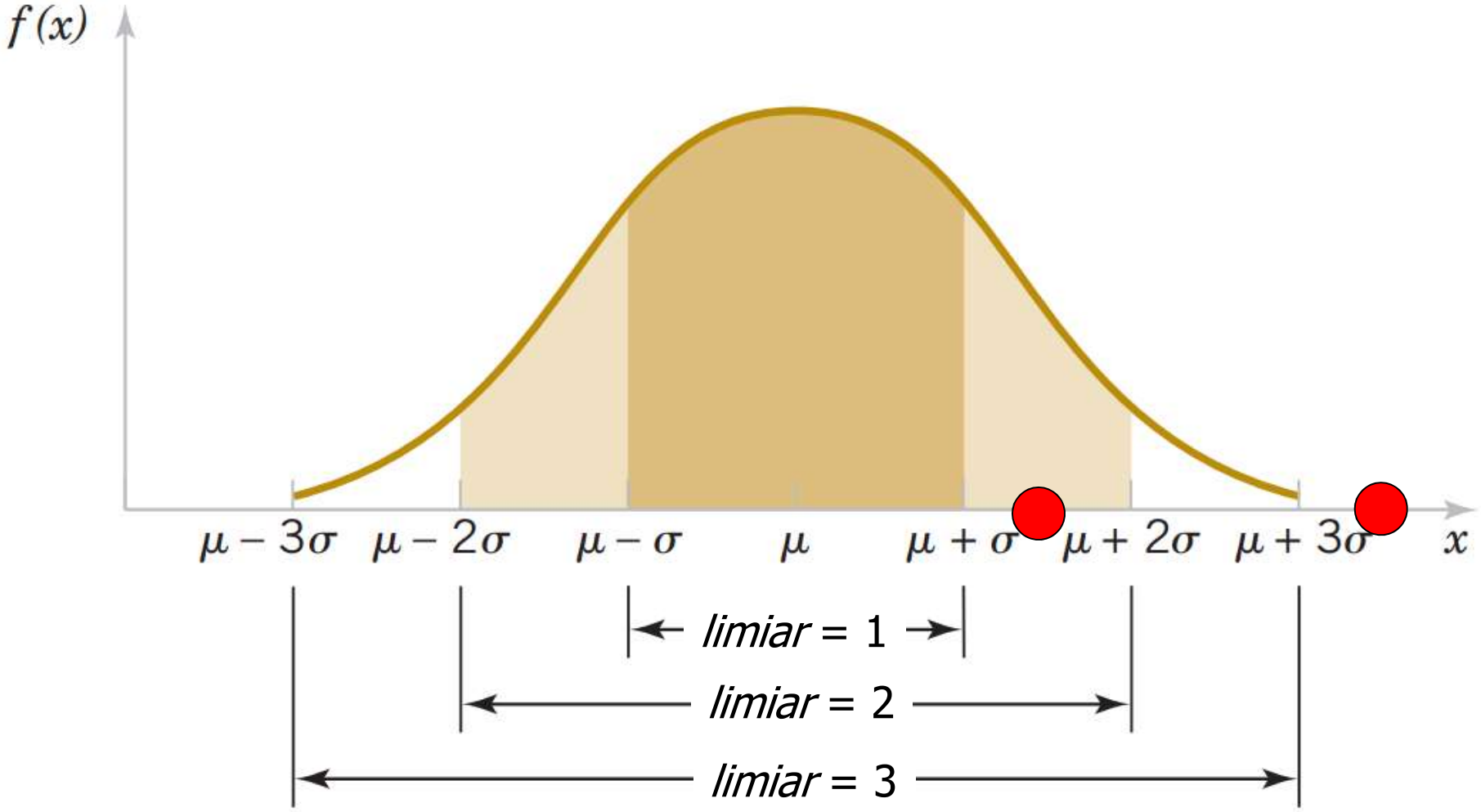
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



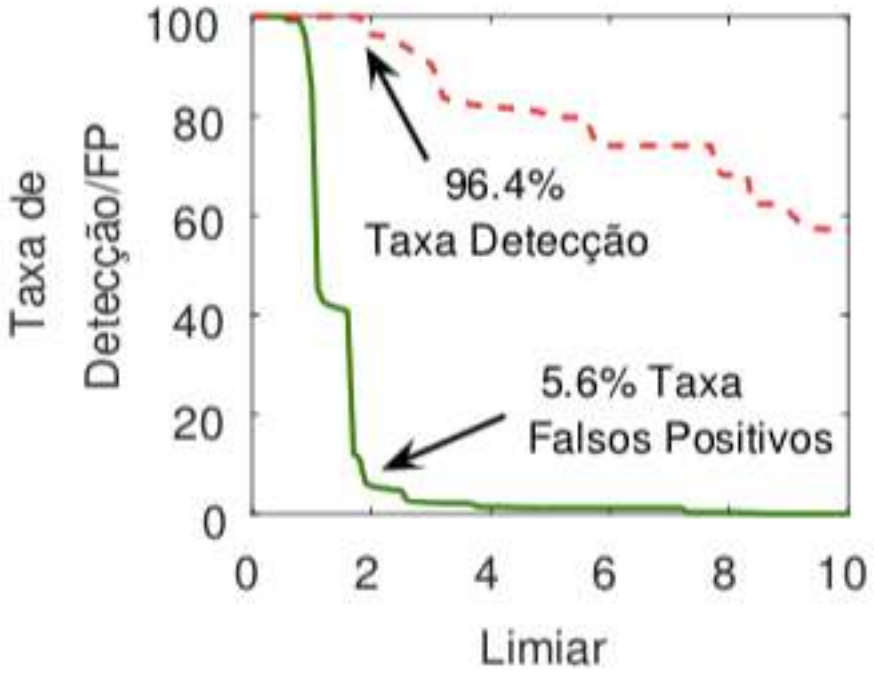
Detecção pela Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

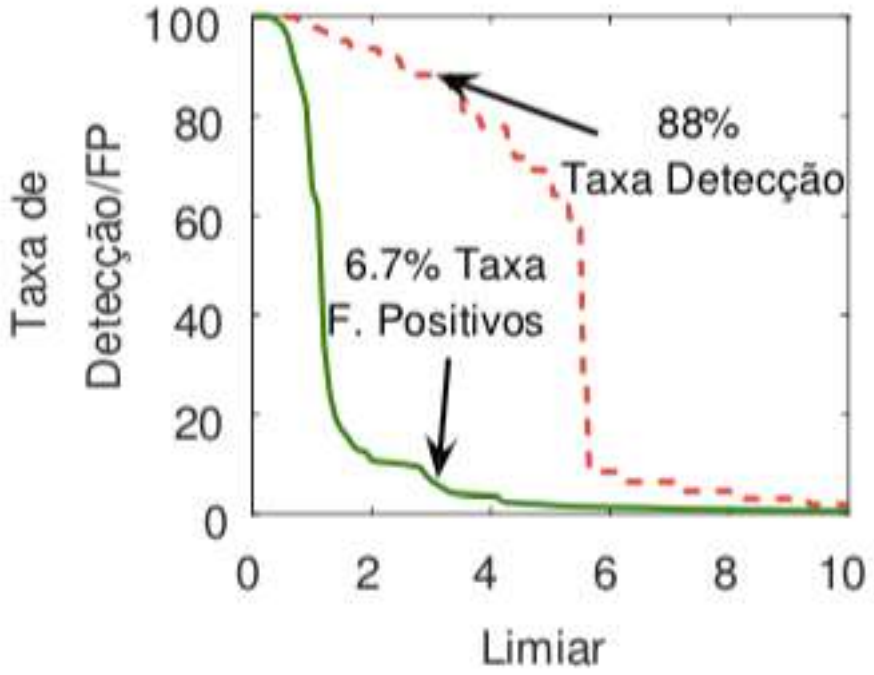


Resultados
Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Conjunto de dados do Laboratório.

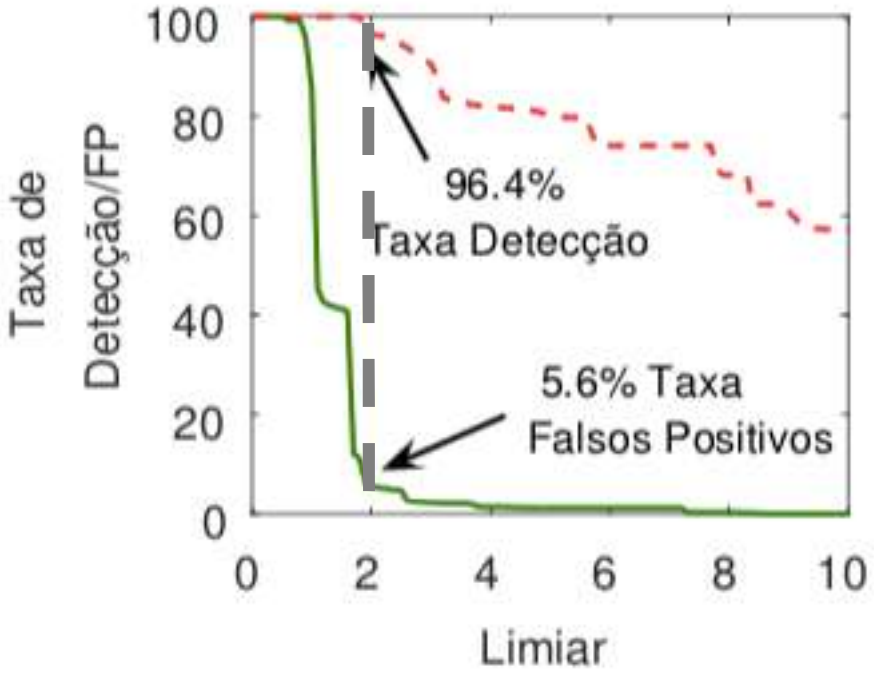


Conjunto de dados NetOp

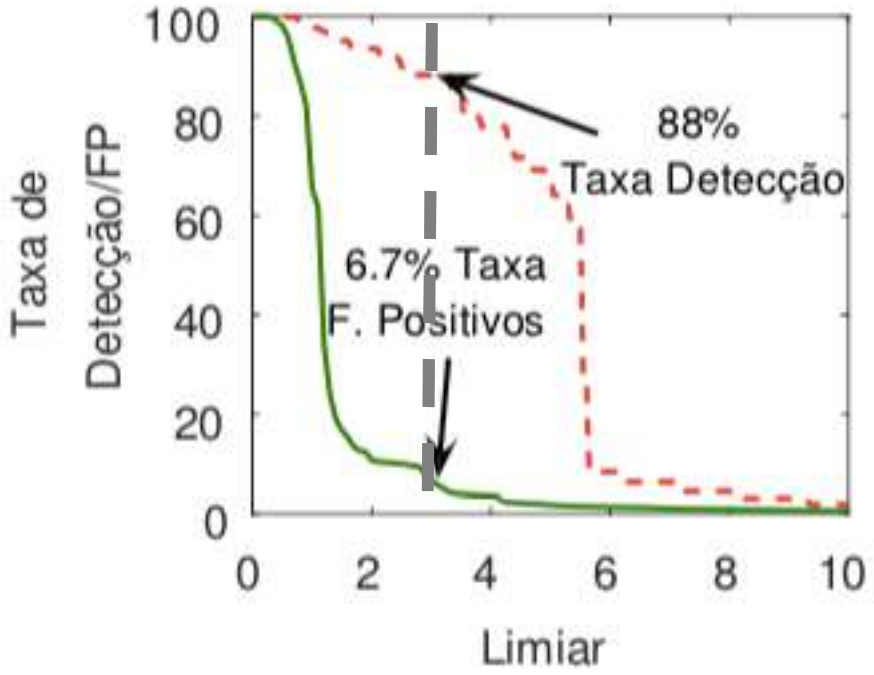
COPPE UFRJ

Resultados
Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Conjunto de dados do Laboratório.



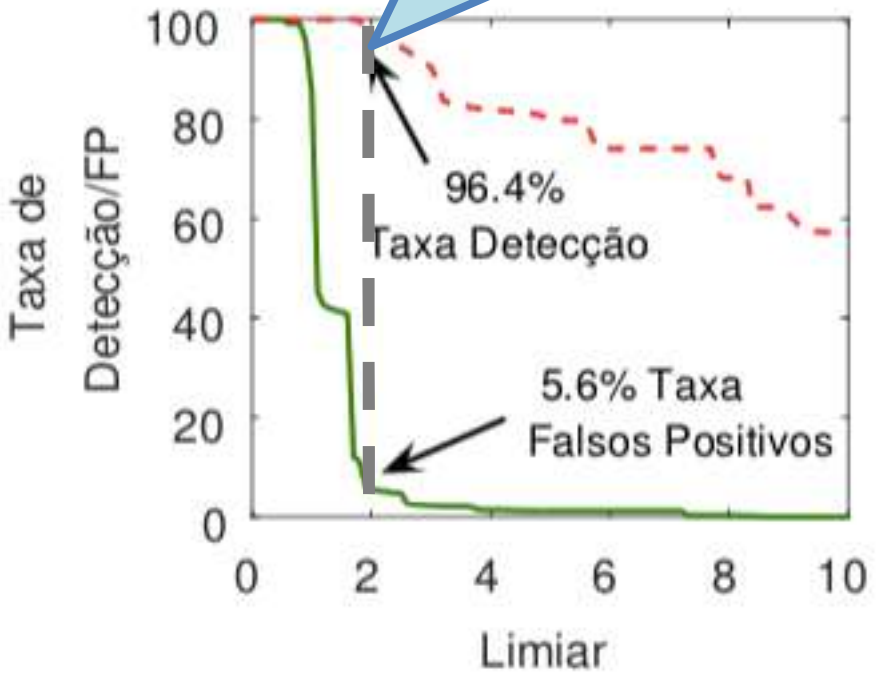
Conjunto de dados NetOp

COPPE UFRJ

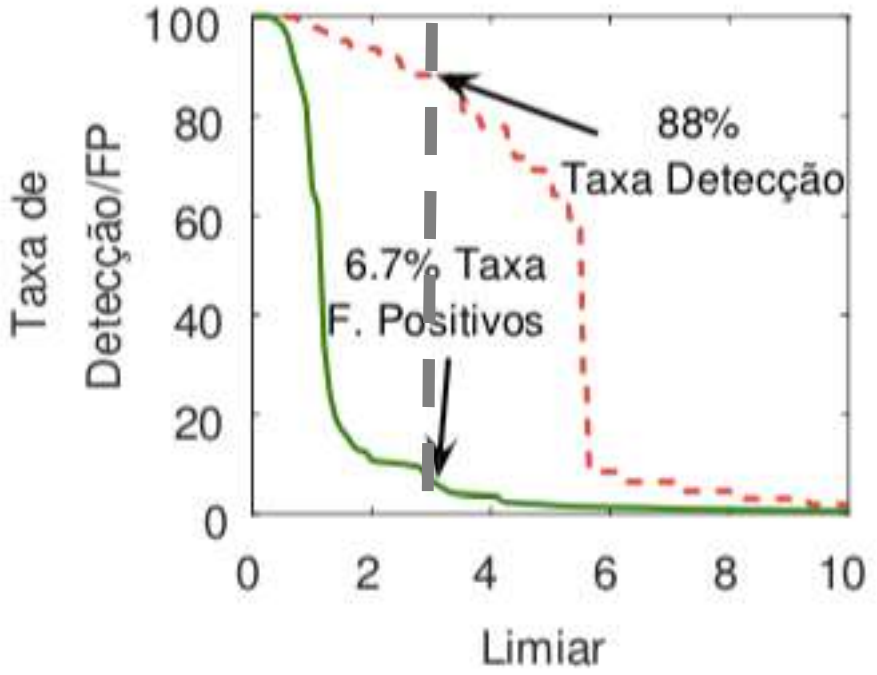
Resultados
Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Limiar = 2
Tráfego com menor variabilidade



Conjunto de dados do Laboratório.

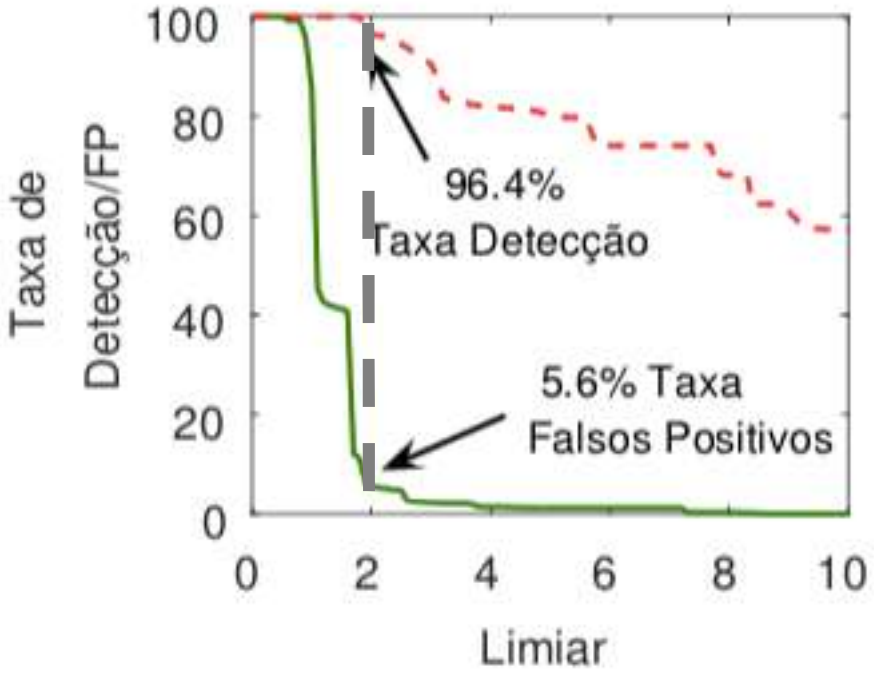


Conjunto de dados NetOp

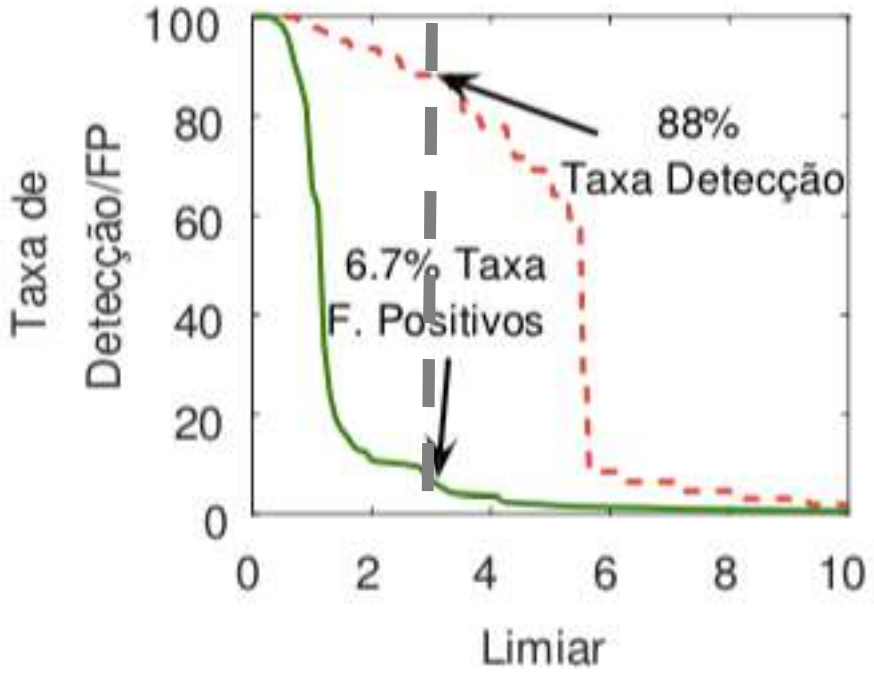
COPPE UFRJ

Resultados
Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Conjunto de dados do Laboratório.

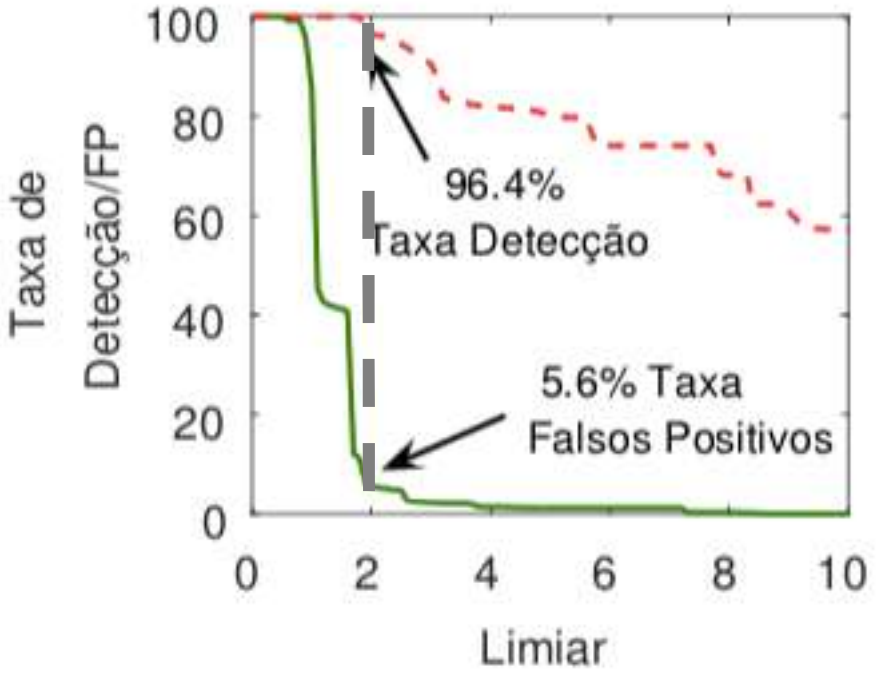


Conjunto de dados NetOp

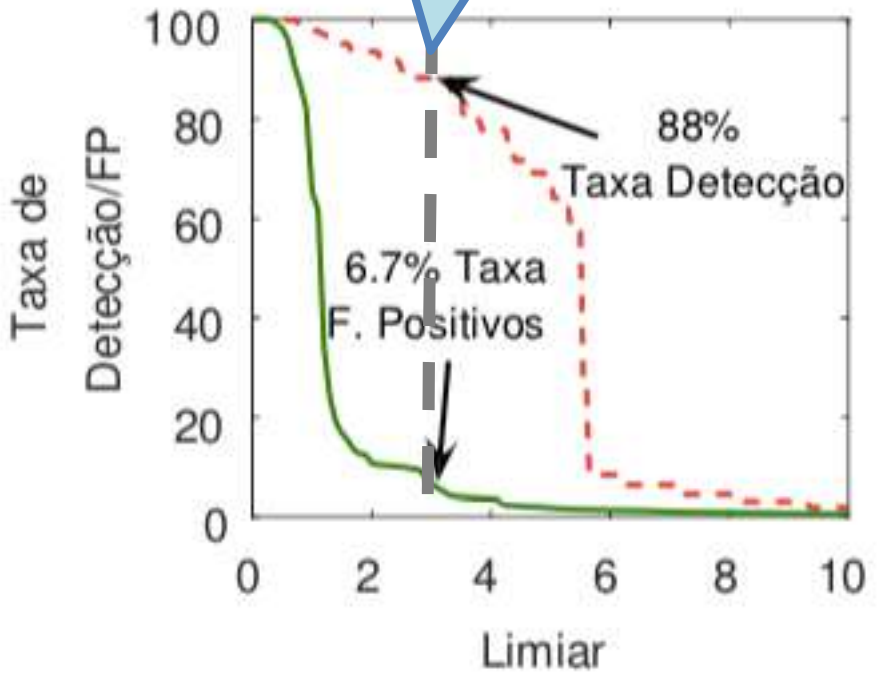
COPPE UFRJ

Resultados
Distribuição Normal

Limiar = 3
Tráfego com maior variabilidade



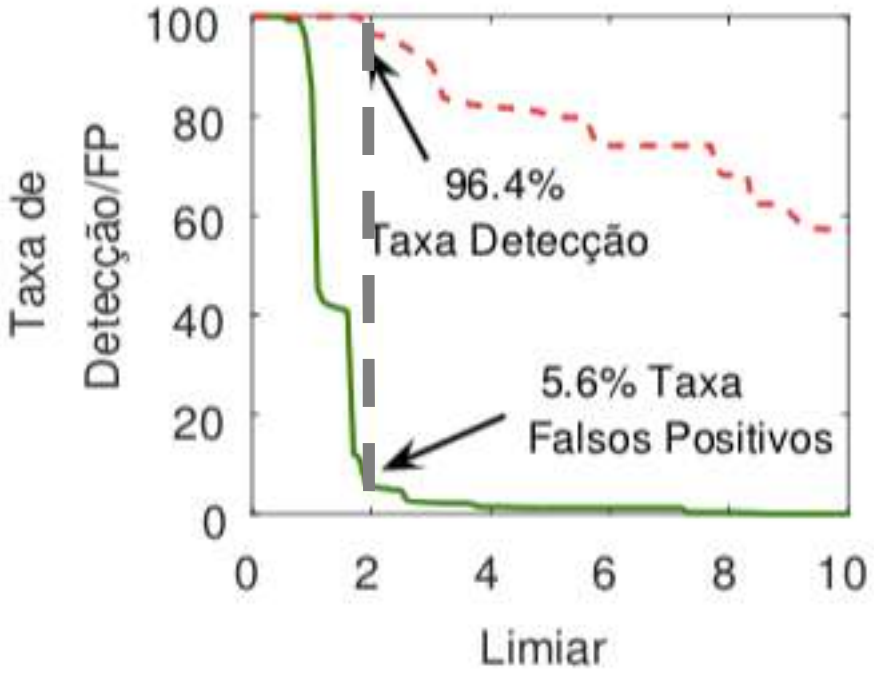
Conjunto de dados do Laboratório.



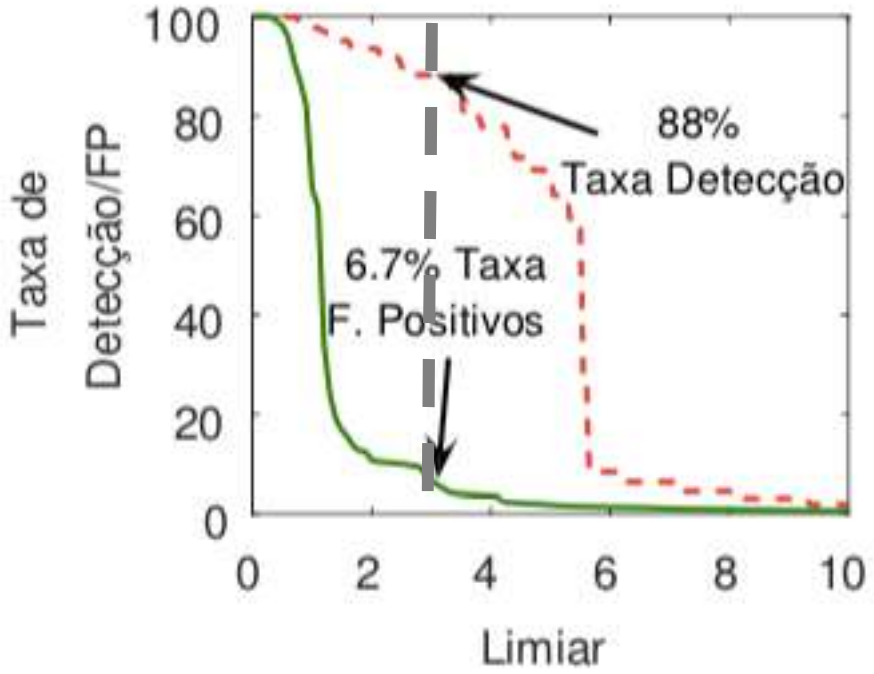
Conjunto de dados NetOp

Resultados
Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Conjunto de dados do Laboratório.

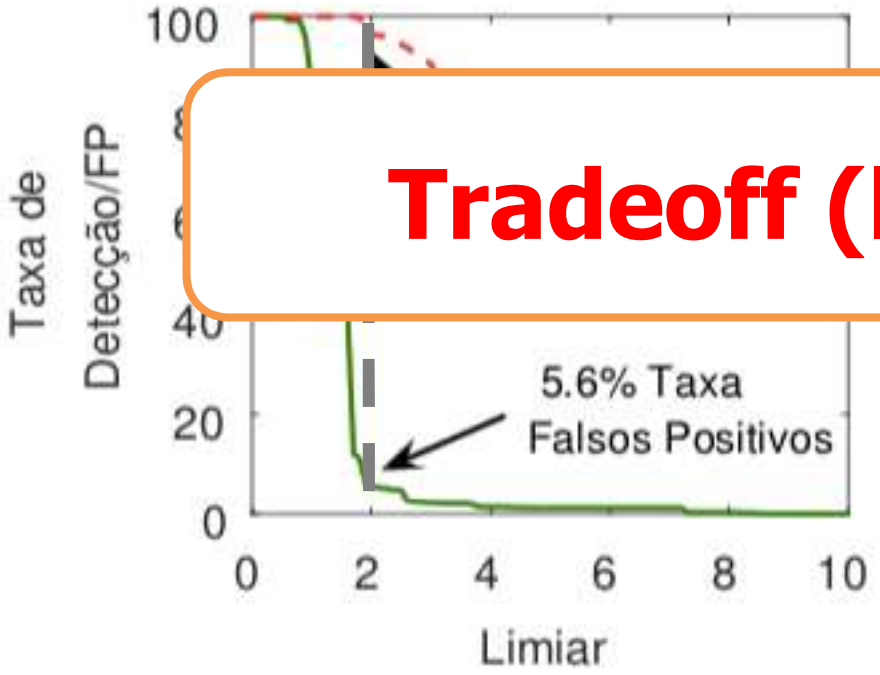


Conjunto de dados NetOp

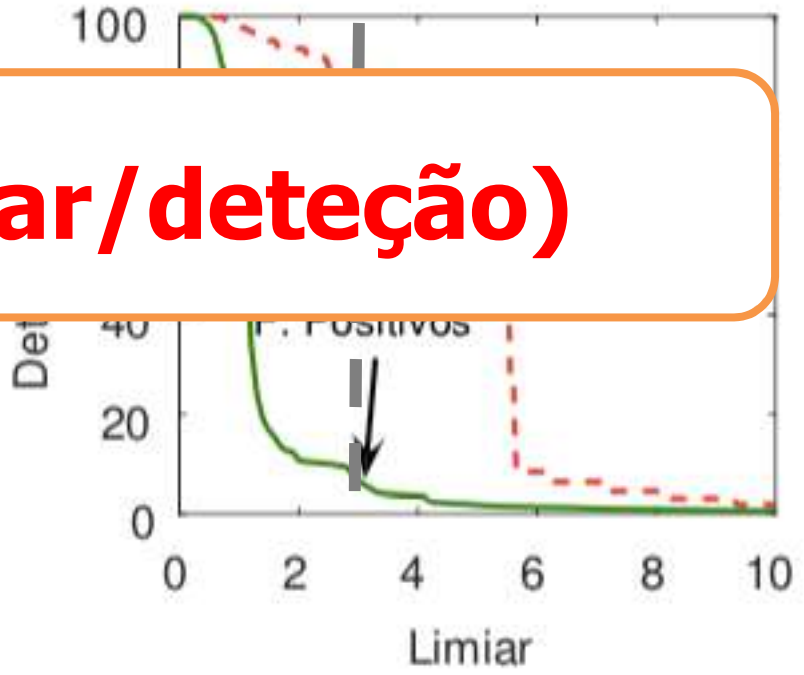
COPPE UFRJ

Resultados
Distribuição Normal

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Tradeoff (limiar/deteção)



Conjunto de dados do Laboratório.

Conjunto de dados NetOp

COPPE UFRJ

Detecção por Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Série Temporal → Janela deslizante de 40 fluxos
- Grau de dispersão das amostras na janela

$$H(X) = -\sum_{i=1}^{N} \left(\frac{n_i}{S}\right) \log_2 \left(\frac{n_i}{S}\right),$$

  - S = número total de observações
  - n_i = número de observações no intervalo i
  - N = número de intervalos

COPPE UFRJ

Detecção por Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Série Temporal → Janela deslizante de 40 fluxos
- Grau de dispersão das amostras na janela

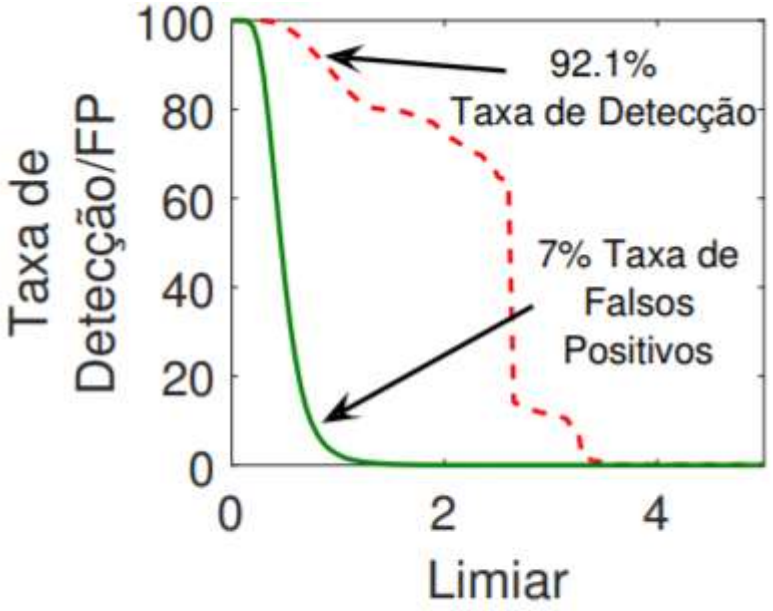
$$H(X) = -\sum_{i=1}^{N} \left(\frac{n_i}{S}\right) \log_2 \left(\frac{n_i}{S}\right),$$

  - S = número total de observações
  - n_i = número de observações no intervalo i
  - N = número de intervalos
- H(X) = zero
  - Valores concentrados em um intervalo
- H(X) = $\log_2(N)$
  - Cada valor esta em uma faixa diferente i

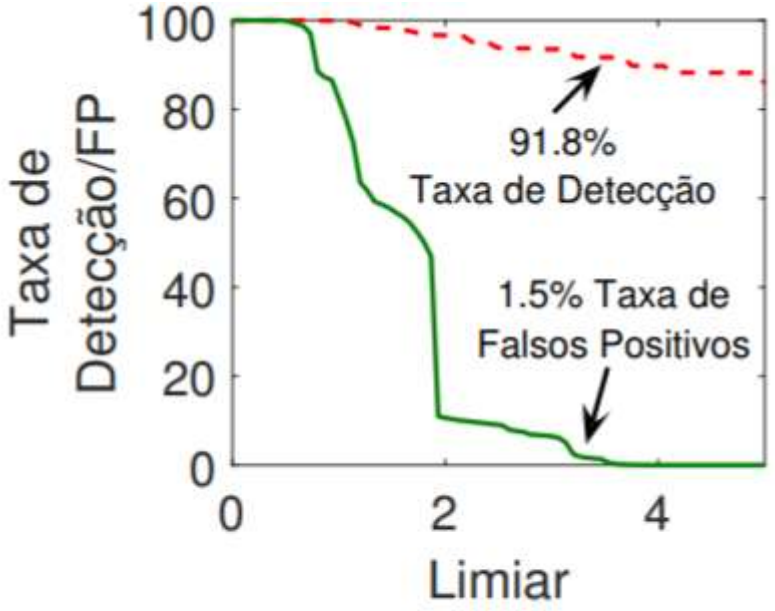
COPPE UFRJ

Resultados
Detecção pela Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



(a) Conjunto de dados do Laboratório.

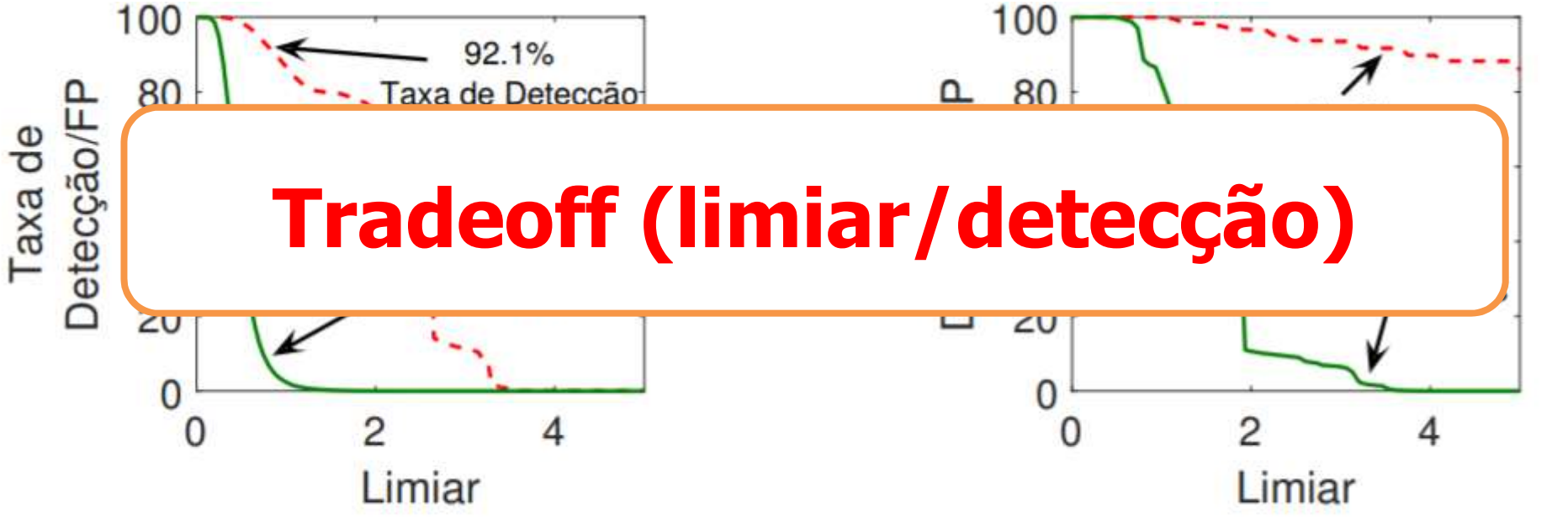


(b) Conjunto de dados do Operador de Rede.

COPPE UFRJ

Resultados
Detecção pela Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Tradeoff (limiar/detecção)

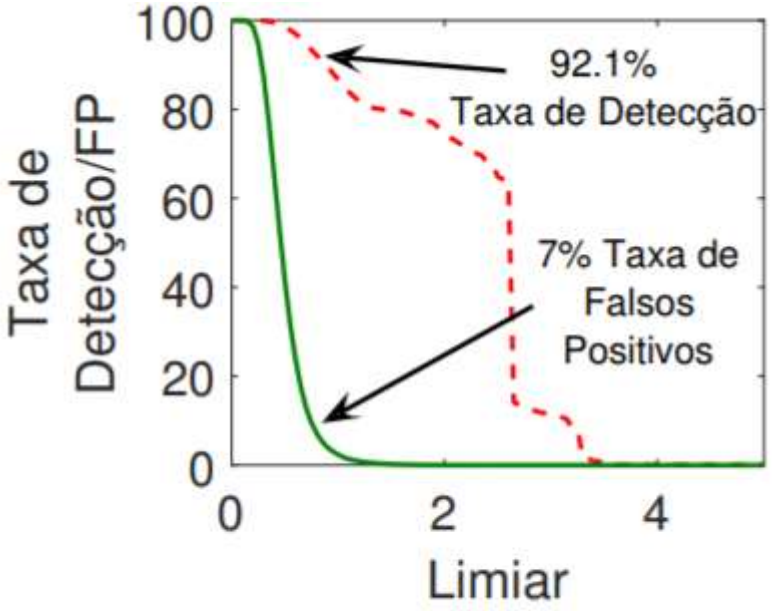
(a) Conjunto de dados do Laboratório.

(b) Conjunto de dados do Operador de Rede.

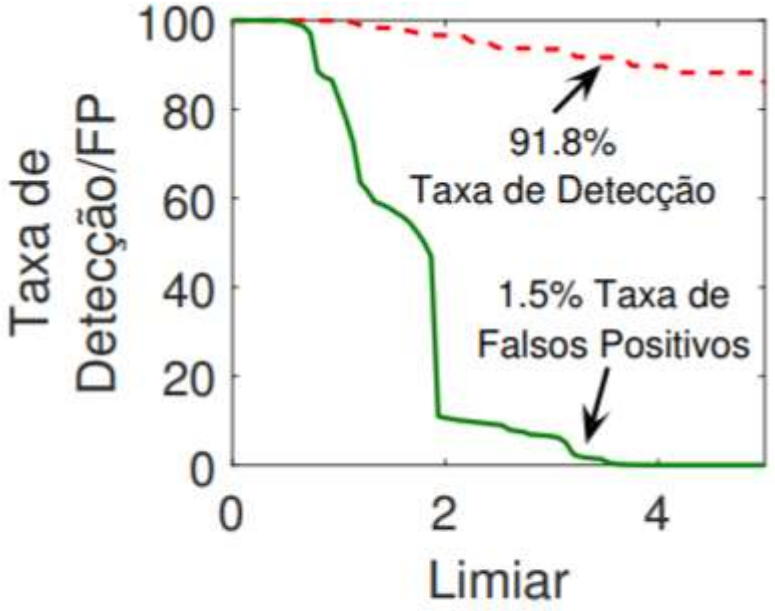
COPPE UFRJ

Resultados
Detecção pela Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



(a) Conjunto de dados do Laboratório.

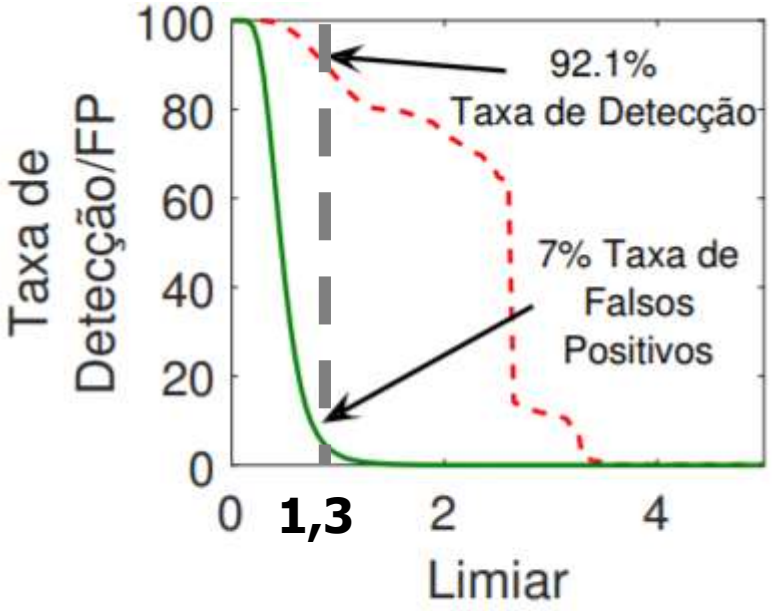


(b) Conjunto de dados do Operador de Rede.

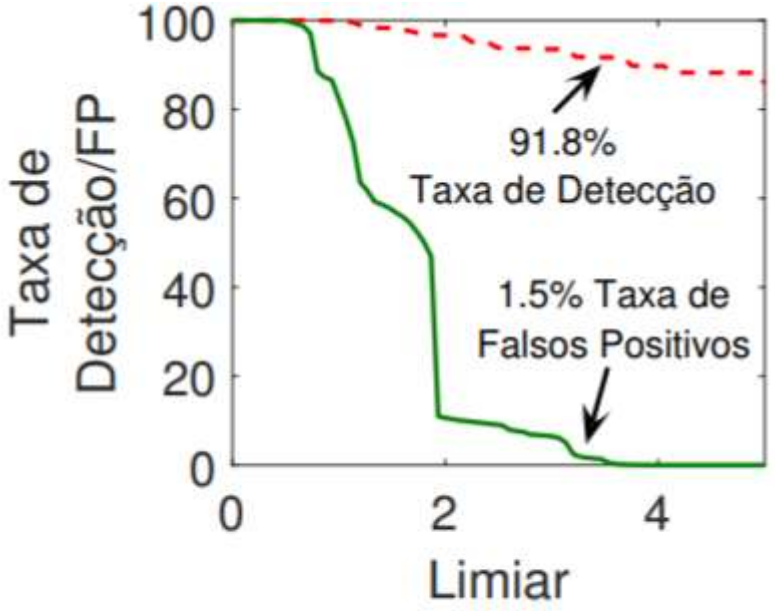
COPPE UFRJ

Resultados
Detecção pela Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



(a) Conjunto de dados do Laboratório.

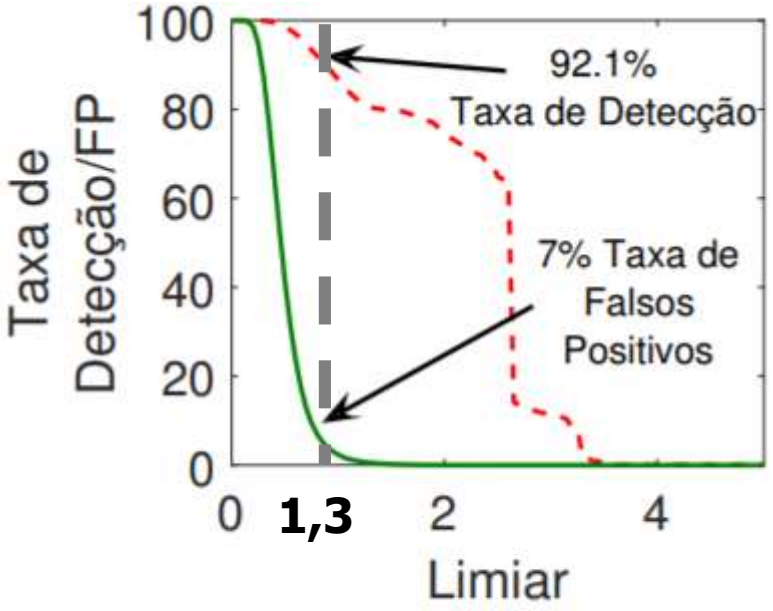


(b) Conjunto de dados do Operador de Rede.

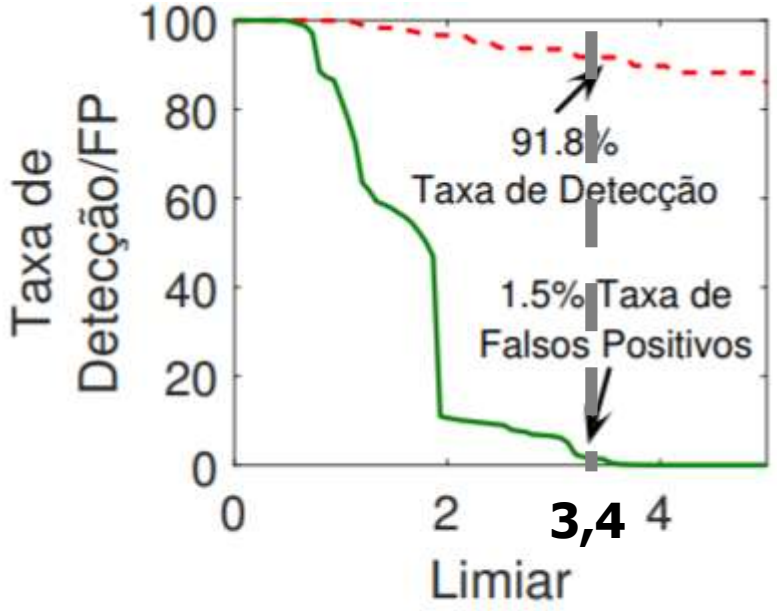
COPPE UFRJ

Resultados
Detecção pela Entropia

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



(a) Conjunto de dados do Laboratório.



(b) Conjunto de dados do Operador de Rede.

COPPE UFRJ

AULA PRÁTICA

Estudo de Caso CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- s**CA**lable **TRA**ffic **C**lassifier and **A**nalyzer
  - Análise de tráfego e visualização através do processamento por fluxo

Estudo de Caso CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- s**CA**lable **TRA**ffic **C**lassifier and **A**nalyzer
  - Análise de tráfego e visualização através do processamento por fluxo
    - *Dashboard* amigável para a visualização
      - Estatísticas de tráfego
      - Comportamento da rede em tempo real

Estudo de Caso CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- s**CA**lable **TRA**ffic **C**lassifier and **A**nalyzer
  - Análise de tráfego e visualização através do processamento por fluxo
    - *Dashboard* amigável para a visualização
      - Estatísticas de tráfego
      - Comportamento da rede em tempo real
  - Classificação por Aprendizado de Máquina
    - Classifição em linha e em tempo diferenciado

Estudo de Caso CATRACA

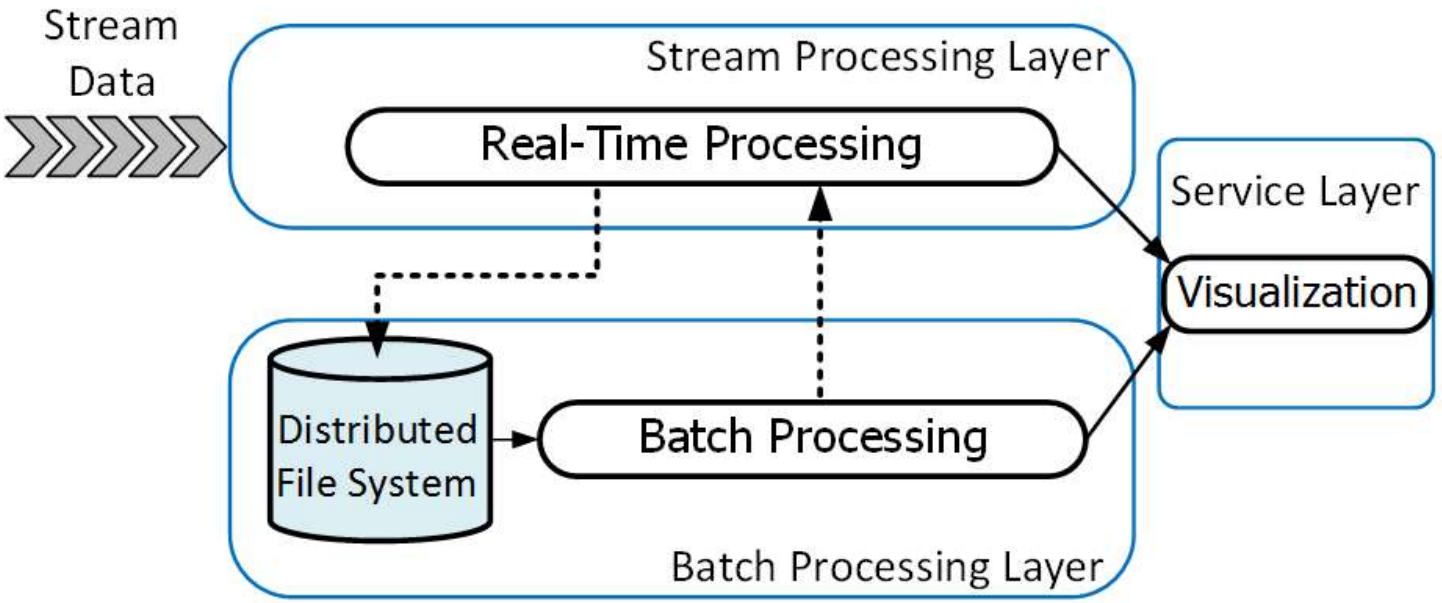
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- s**CA**lable **TRA**ffic **C**lassifier and **A**nalyzer
  - Análise de tráfego e visualização através do processamento por fluxo
    - *Dashboard* amigável para a visualização
      - Estatísticas de tráfego
      - Comportamento da rede em tempo real
  - Classificação por Aprendizado de Máquina
    - Classifição em linha e em tempo diferenciado
  - Escalabilidade e Processamento por Fluxo
    - Implemented in OPNFV
    - Apache Spark

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

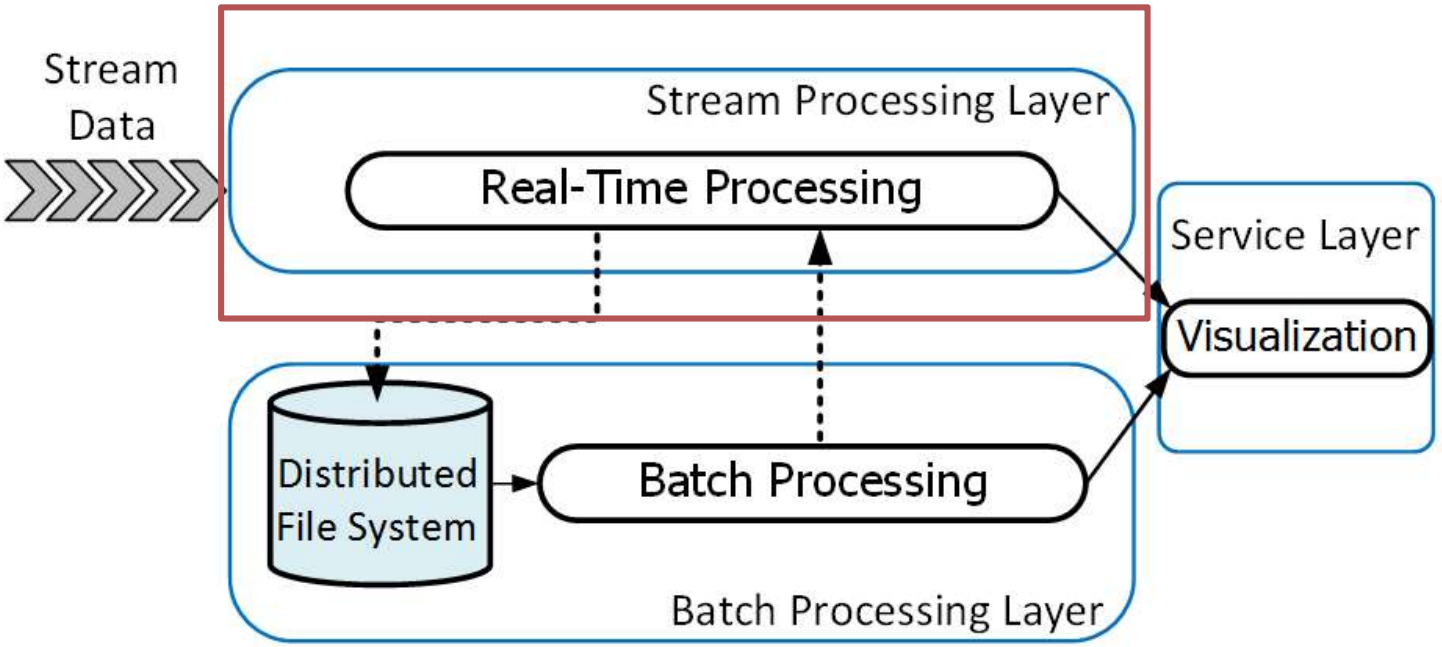


COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

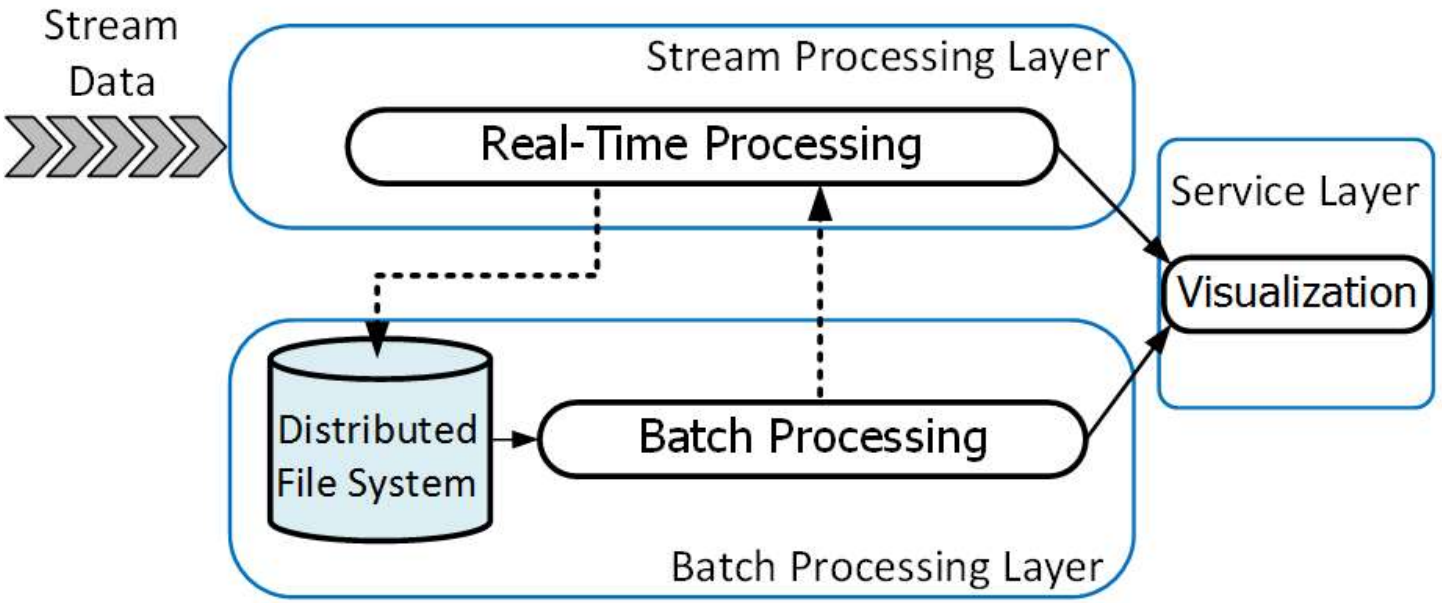


COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

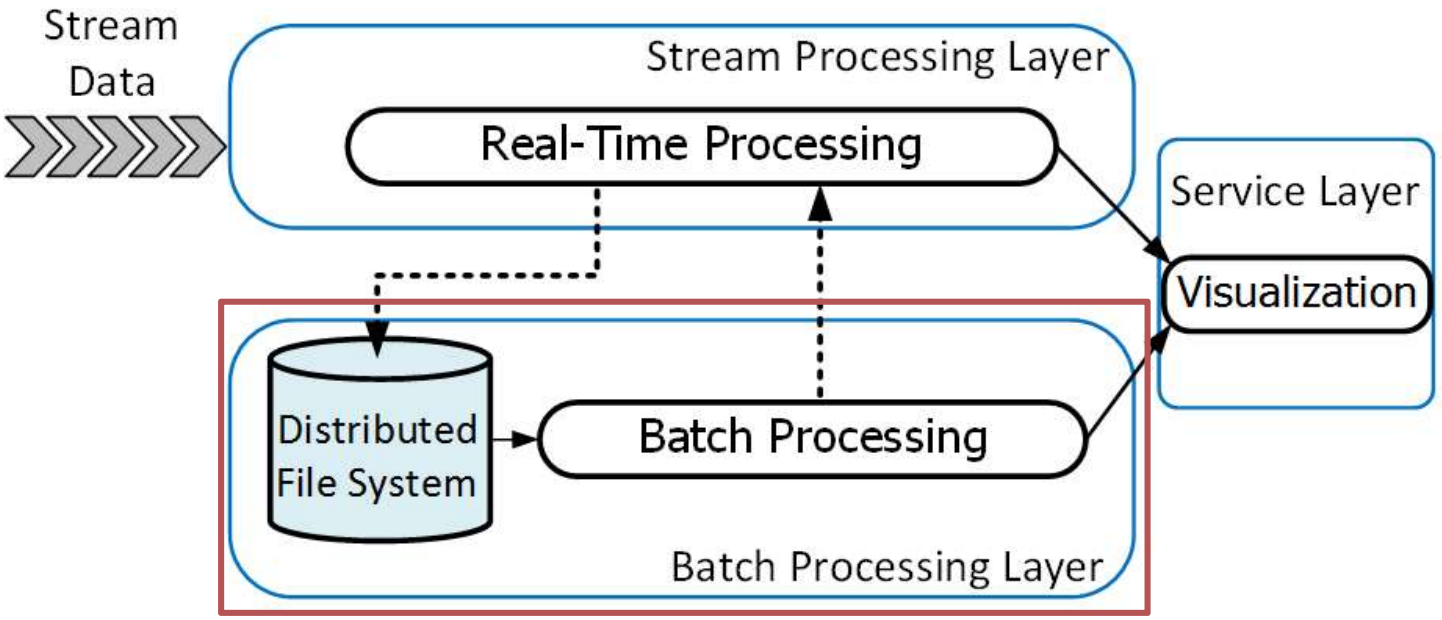


COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

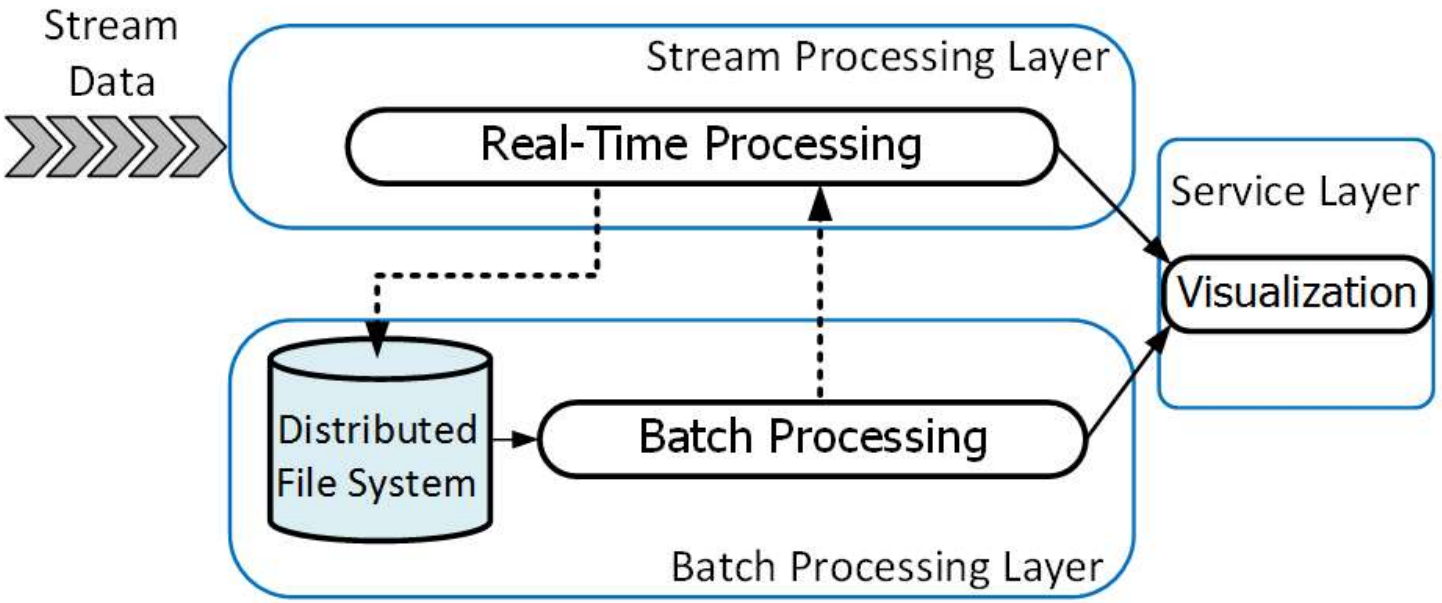


COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

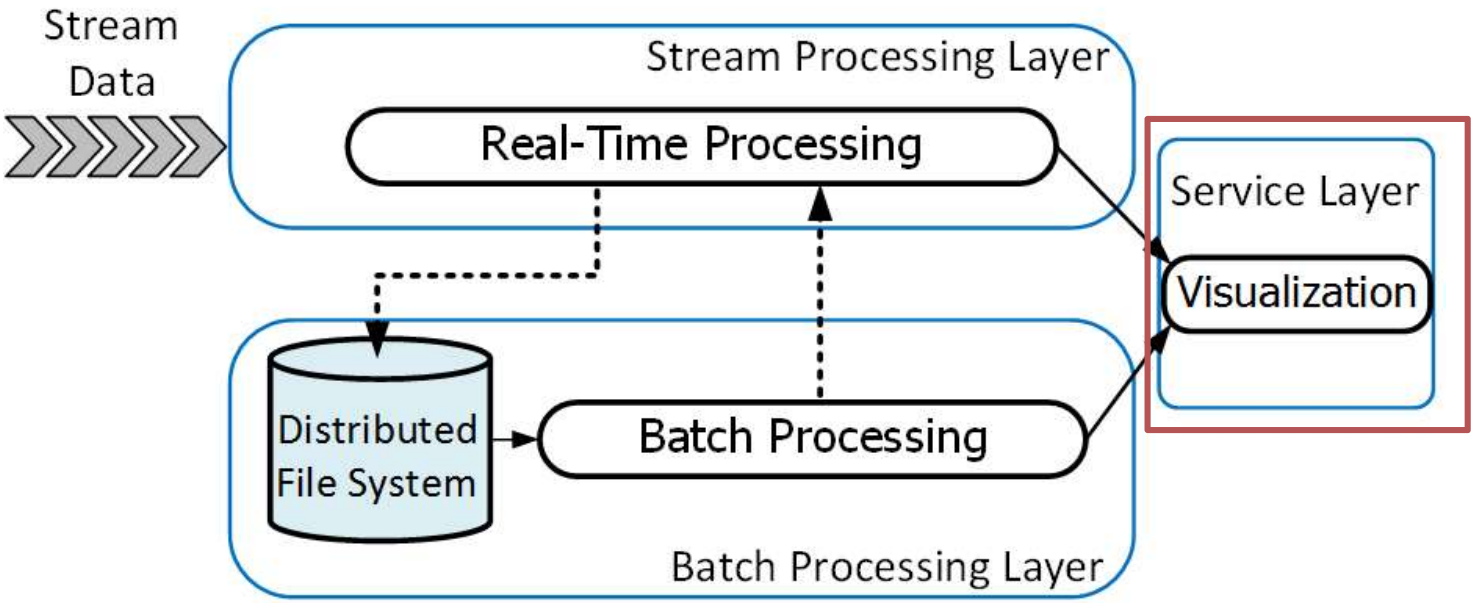


COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

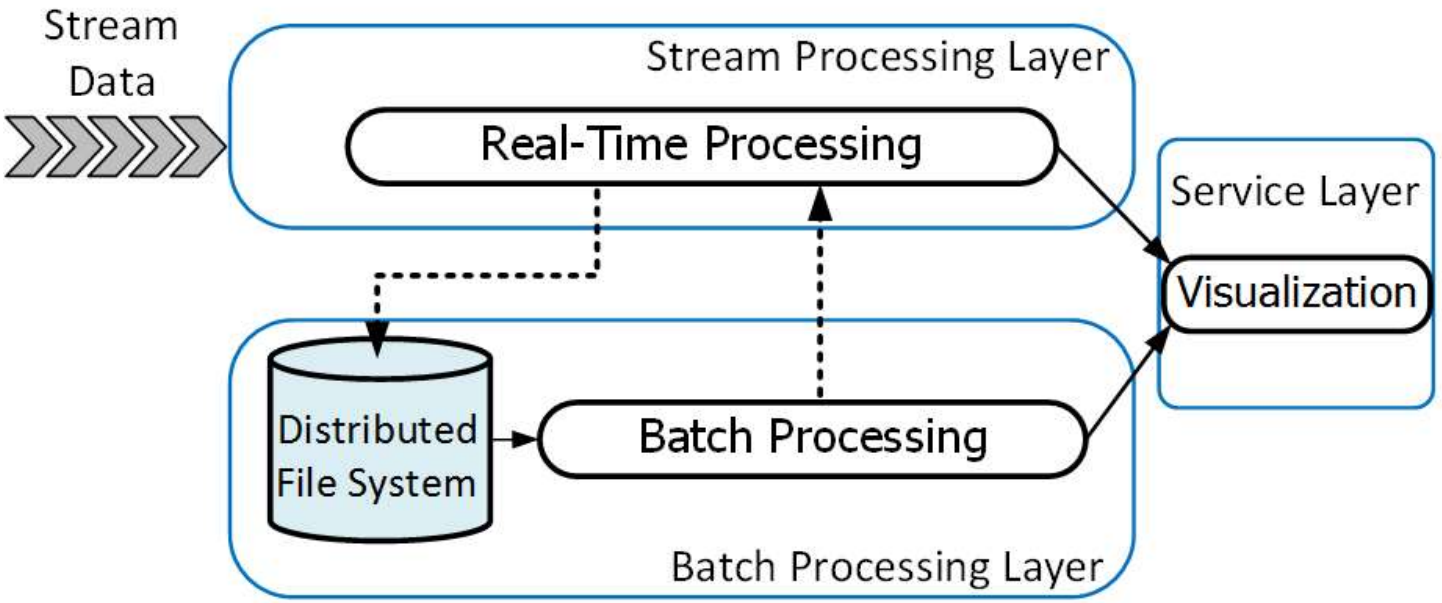


COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

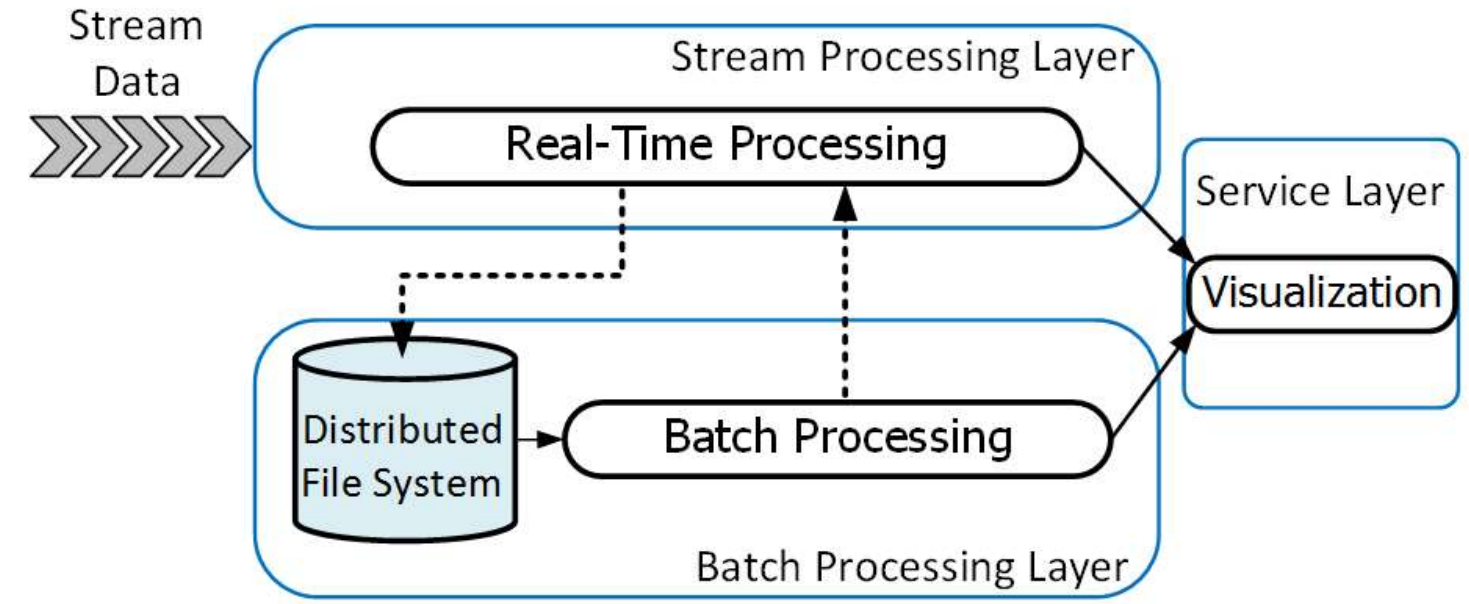


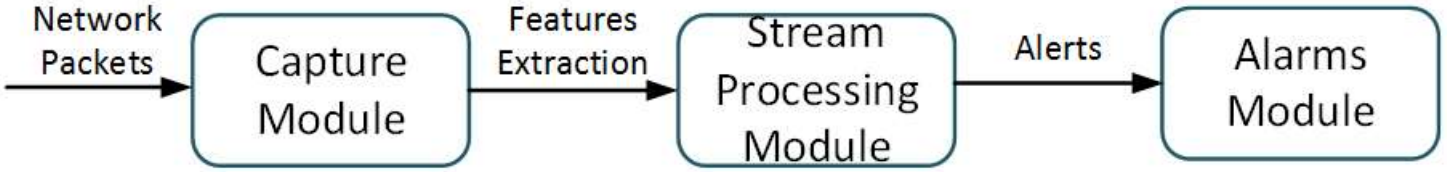
COPPE UFRJ

Abordagem Proposta CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Baseada na Arquitetura Lambda

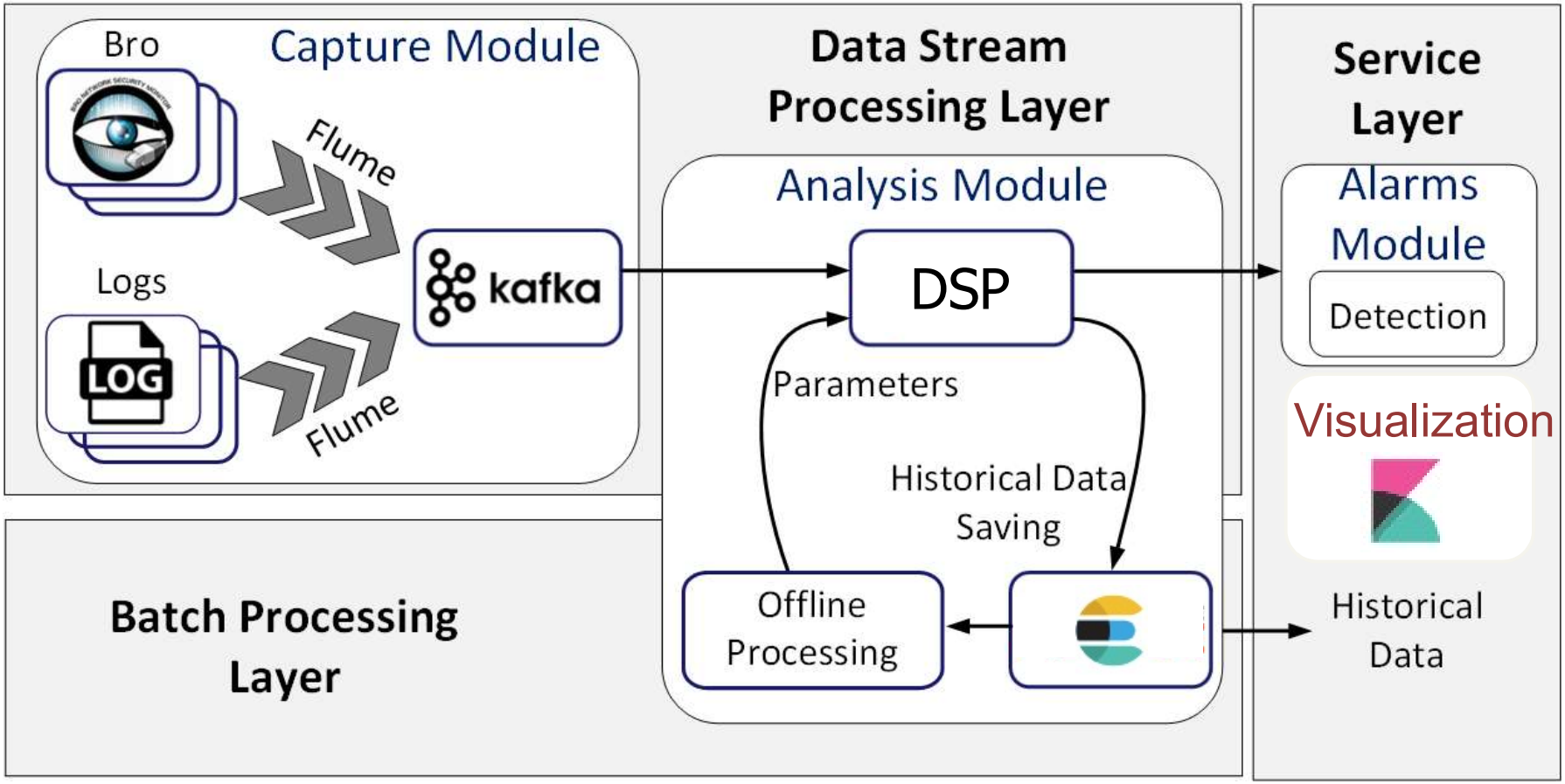




COPPE UFRJ

Arquitetura do Sistema CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

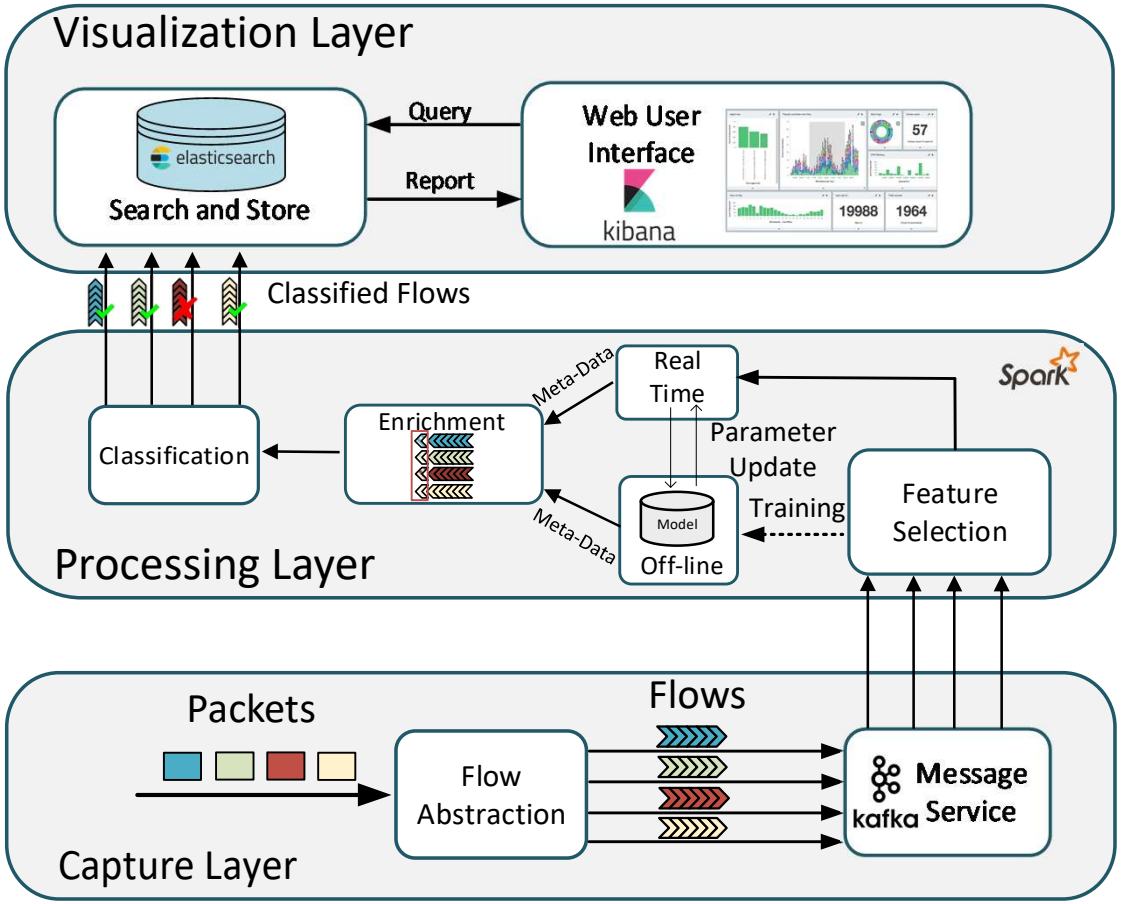
Roteiro da Aula Prática CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Arquitetura em camadas
  - Relacionamento entre cada componente e aplicação da arquitetura *lambda*
- Código executado em cada camada
  - Explicação do funcionamento
- Integração das camadas
  - Como são executados cada componente das camadas
- Demonstração da execução da ferramenta CATRACA
  - Execução no agrupamento computacional no GTA/UFRJ

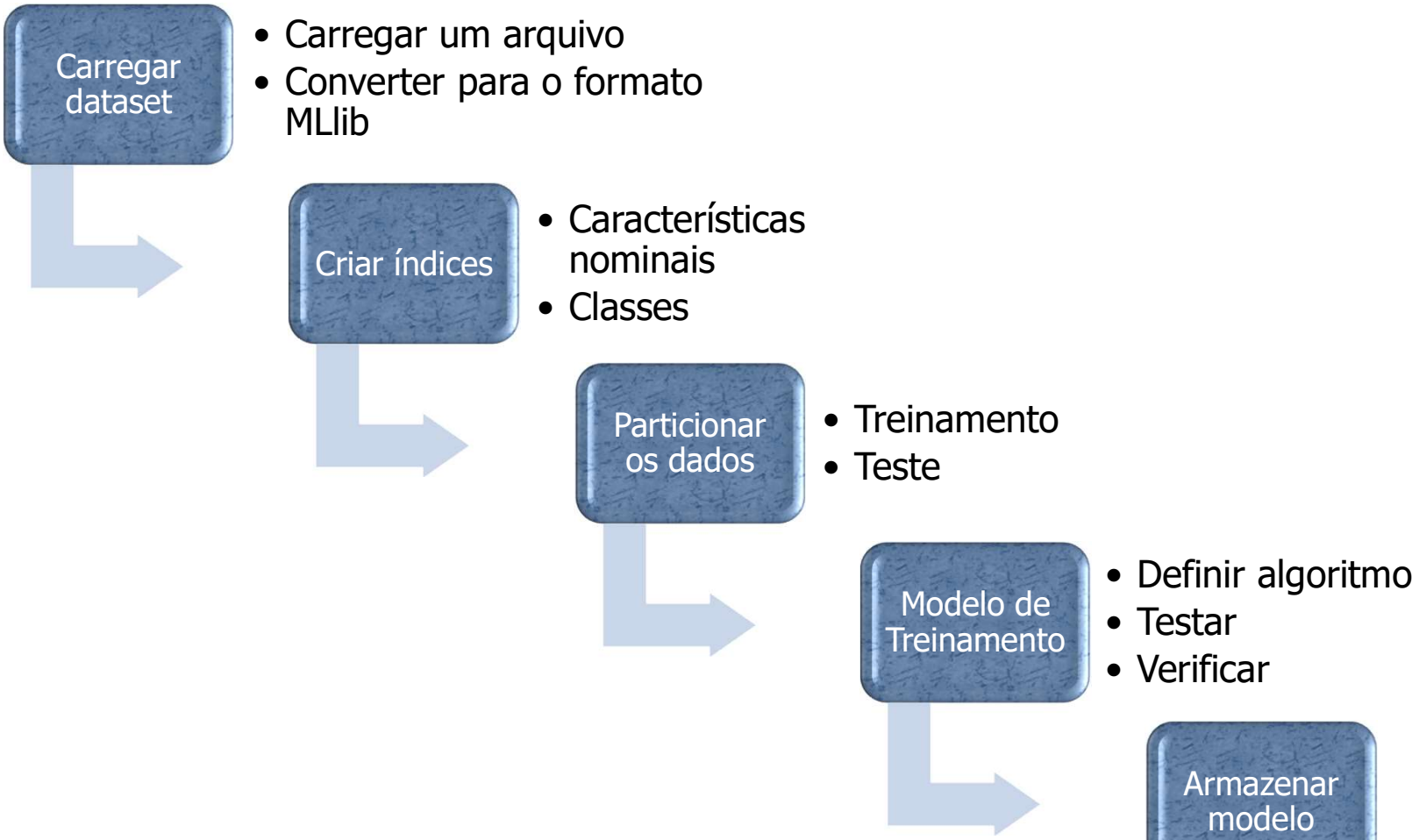
Arquitetura do Sistema CATRACA → Mais Detalhes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Roteiro de Treinamento Offline



Bibliotecas

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Importar bibliotecas de aprendizado de máquina do Spark

```
from __future__ import print_function

import sys

from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import StringIndexer, VectorIndexer, VectorAssembler
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
```

Intergindo com o Spark

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Spark session
  - Objeto que controla a interação do nó mestre com o restante do aglomerado

```
if __name__ == "__main__":
    # Create spark session
    spark = SparkSession\
        .builder\
        .appName("TrainDecisionTreeIDS")\
        .getOrCreate()
```


COPPE UFRJ

Carregando o Dataset

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Definir o cabeçalho (schema) dos dados

```
schema = StructType([
        StructField("srcip", StringType(), False),              # Feature 1
        StructField("srcport", IntegerType(), False),           # Feature 2
        StructField("dstip", StringType(), False),              # Feature 3
        StructField("dstport", IntegerType(), False),           # Feature 4
        StructField("proto", IntegerType(), False),             # Feature 5
        #(...)
        StructField("burg_cnt", IntegerType(), False),          # Feature 42
        StructField("total_fhlen", IntegerType(), False),       # Feature 43
        StructField("total_bhlen", IntegerType(), False),       # Feature 44
        StructField("dscp", IntegerType(), False),              # Feature 45
        StructField("label", StringType(), False)               # Class Label
    ])
```


COPPE UFRJ

Carregando o Dataset

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Carregar os dados de um arquivo

```
data = spark.read.csv("File.csv", schema=schema)
# or
data = spark.read.json("File.json", schema=schema)

# For more formats:
# http://spark.apache.org/docs/2.1.0/api/python/pyspark.sql.html
```


COPPE UFRJ

Transformando para o Formato MLlib

- Dividir em duas colunas
  - Vetores de Características
  - Classes

```
# Create vector assembler
assembler = VectorAssembler(inputCols=schema.fieldNames()[:-1],
                            outputCol="features")

# Assemble feature vector in new dataframe
assembledData = assembler.transform(data)
```

Criando um Modelo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Definir um classificador

```
# Create a DecisionTree model trainer
dt = DecisionTreeClassifier(labelCol="indexedLabel",
                            featuresCol="indexedFeatures")

# Train a RandomForest model.
rf = RandomForestClassifier(labelCol="indexedLabel",
                            featuresCol="indexedFeatures",
                            numTrees=10)

# For more models, look into
# https://spark.apache.org/docs/latest/ml-classification-regression.html
```


COPPE UFRJ

Particionando os Dados

- Gerar novos conjuntos de dados
  - 80% treino
  - 20% teste

```
(trainingData, testData) = assembledData.randomSplit([0.8, 0.2])
```

Treinando o Modelo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Criar um pipeline que encadeia as transformações
  - Transormações retornam um novo dataset
- Ajustar o modelo de acordo com os dados de treinamento e testar o modelo

```
# Chain indexers and model training in a Pipeline
pipeline = Pipeline(stages=[labelIndexer, featureIndexer, dt])

# Train model
model = pipeline.fit(trainingData)

# Make predictions
predictions = model.transform(testData)
```


COPPE UFRJ

Verificando Modelo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Um avaliador retorna métricas a partir dos testes

```
f1=MulticlassClassificationEvaluator(labelCol="indexedLabel",
             predictionCol="prediction",
             metricName="f1").evaluate(predictions)
Precision=MulticlassClassificationEvaluator(labelCol="indexedLabel",
             predictionCol="prediction",
             metricName="weightedPrecision").evaluate(predictions)
Recall=MulticlassClassificationEvaluator(labelCol="indexedLabel",
            predictionCol="prediction",
            metricName="weightedRecall").evaluate(predictions)
accuracy=MulticlassClassificationEvaluator(labelCol="indexedLabel",
            predictionCol="prediction",
            metricName="accuracy").evaluate(predictions)
```


COPPE UFRJ

Armazenando Modelo

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

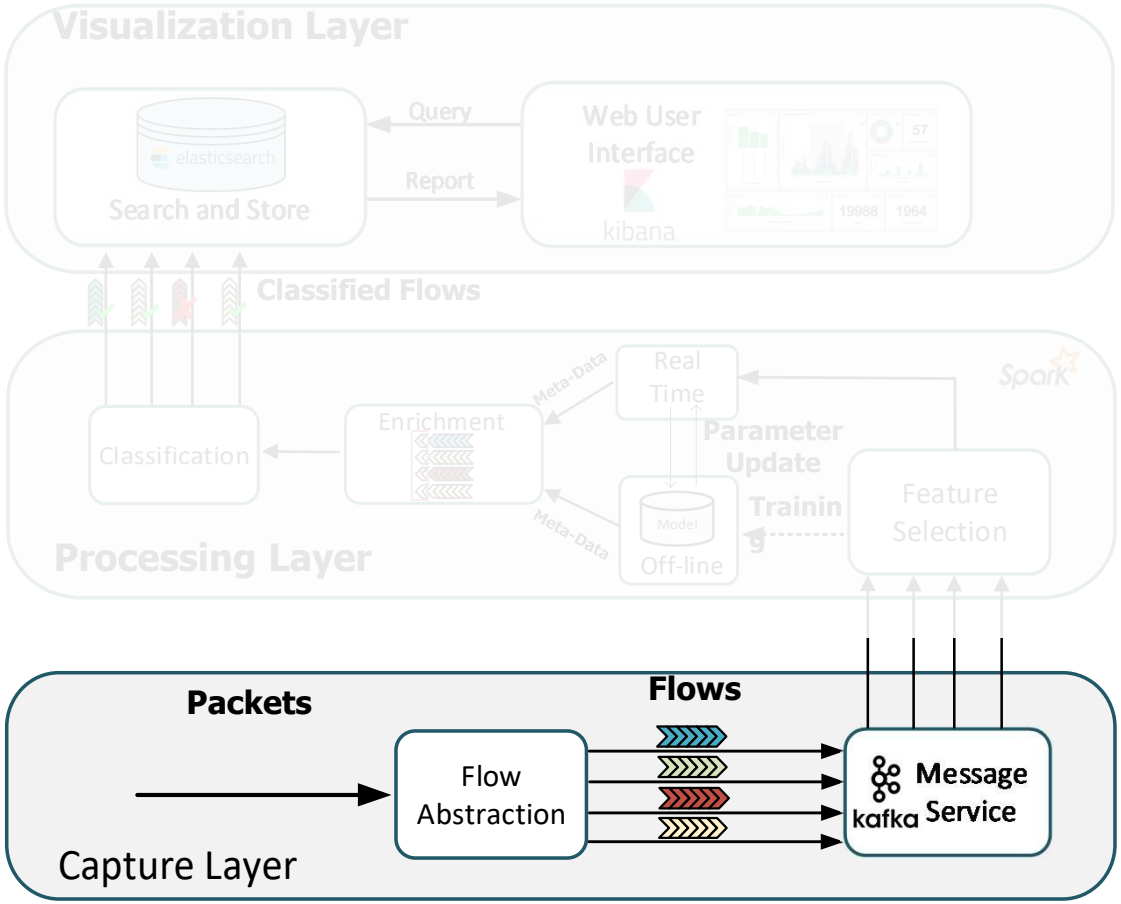
- O modelo é armazenado em um diretório

```
# Store model
model.save("MyModelPath")
```


COPPE UFRJ

CATRACA
Camada de Captura

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

CATF
Car

TA / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

```
import threading
import Queue
import os
import netifaces as ni
import flowtbag
import sys
import json

#If Windows
if os.name == 'nt': import win_inet_pton

from scapy.all import *
load_contrib("nsh")

from kafka import KafkaProducer
from kafka.errors import KafkaError

###Config###
WINDOW = 2.0
CLASS = 'live'
TEST_FILE = 'testFlowtbag.txt'
IFACE = 'ens3' #IFACE = 'eth0' #None = All
TOPIC = 'test2'.encode('utf-8')# + str(contadorTopicos)
ENCAPSULATION = None #GRE #NSH
producer = KafkaProducer(bootstrap_servers=['10.240.180.33:9092'])

############

#CallBack
def removeNSH(pkt):
    global PKTS
    global ENCAPSULATION
    try:
        PKTS.append(pkt[ENCAPSULATION][1])
    except:
        pass

def noNSH(pkt):
    global PKTS
    try:
```

COPPE UFRJ

CATF
Ca

```
import threading
import Queue
import os
import netifaces as ni
import flowtbag
import sys
import json

#If Windows
if os.name == 'nt': import win_inet_pton

from scapy.all import *
load_contrib("nsh")

from kafka import KafkaProducer
from kafka.errors import KafkaError

###Config###
WINDOW = 2.0
CLASS = 'live'
TEST_FILE = 'testFlowtbag.txt'
IFACE = 'ens3' #IFACE = 'eth0' #None = All
TOPIC = 'test2'.encode('utf-8')# + str(contadorTopicos)
ENCAPSULATION = None #GRE #NSH
producer = KafkaProducer(bootstrap_servers=['10.240.180.33:9092'])

############

#CallBack
def removeNSH(pkt):
    global PKTS
    global ENCAPSULATION
    try:
        PKTS.append(pkt[ENCAPSULATION][1])
    except:
        pass

def noNSH(pkt):
    global PKTS
    try:
```

Flowtbag → Pacote usado para abstrair pacotes de rede em conjunto de características por fluxos (5-tupla)

COPPE UFRJ

CATF
Car

TA / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

```
import threading
import Queue
import os
import netifaces as ni
import flowtbag
import sys
import json

#If Windows
if os.name == 'nt': import win_inet_pton

from scapy.all import *
load_contrib("nsh")

from kafka import KafkaProducer
from kafka.errors import KafkaError

###Config###
WINDOW = 2.0
CLASS = 'live'
TEST_FILE = 'testFlowtbag.txt'
IFACE = 'ens3' #IFACE = 'eth0' #None = All
TOPIC = 'test2'.encode('utf-8')# + str(contadorTopicos)
ENCAPSULATION = None #GRE #NSH
producer = KafkaProducer(bootstrap_servers=['10.240.180.33:9092'])

############

#CallBack
def removeNSH(pkt):
    global PKTS
    global ENCAPSULATION
    try:
        PKTS.append(pkt[ENCAPSULATION][1])
    except:
        pass

def noNSH(pkt):
    global PKTS
    try:
```

COPPE UFRJ

Configurações gerais:
Tempo da janela de amostragem, interface de captura, tópico do Kafka (serviço de mensagens), Protocolo de encapsulamento usado no enlace

```
from ka        KafkaProducer
from kafk      rs import KafkaError

###Config###
WINDOW = 2.0
CLASS = 'live'
TEST_FILE = 'testFlowtbag.txt'
IFACE = 'ens3' #IFACE = 'eth0' #None = All
TOPIC = 'test2'.encode('utf-8')# + str(contadorTopicos)
ENCAPSULATION = None #GRE #NSH
producer = KafkaProducer(bootstrap_servers=['10.240.180.33:9092'])

############

#CallBack
def removeNSH(pkt):
    global PKTS
    global ENCAPSULATION
    try:
        PKTS.append(pkt[ENCAPSULATION][1])
    except:
        pass

def noNSH(pkt):
    global PKTS
    try:
```

CATP
Car

A / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

```
    try:
        PKTS.append(pkt)
    except:
        pass

#Send to Kafka
def salvar_fluxosKafka(flows, classe, topico, producer):
    for flow in flows.active_flows.values():
        msg = str(flow) + ',' + classe
        producer.send(topico, json.dumps(msg).encode('utf-8'))
BUFFER = Queue.Queue()
PKTS = []
def capture():
    global BUFFER
    global PKTS
    ip = ni.ifaddresses(IFACE)[2][0]['addr']
    while(1):
        #Network to flowtbag format
        PKTS = []
        sniff(timeout=WINDOW, iface=IFACE, store=0, prn=noNSH)
        pkts = [(len(pkt), str(pkt), pkt.time) for pkt in PKTS]
        BUFFER.put(pkts)

def write_flows():
    global BUFFER
    global TOPIC
    global producer
    while (1):
        pkts = BUFFER.get()
        flows=flowtbag.Flowtbag(pkts)
        salvar_fluxosKafka(flows, CLASS, TOPIC, producer)

#Start simple threads
capture_thread = threading.Thread(target=capture)
writer_thread = threading.Thread(target=write_flows)

capture_thread.start()
writer_thread.start()
```

COPPE UFRJ

CATF
Car

```
    try:
        PKTS.append(pkt)
    except:
        pass

#Send to Kafka
def salvar_fluxosKafka(flows, classe, topico, produc
    for flow in flows.active_flows.values():
        msg = str(flow) + ',' + classe
        producer.send(topico, json.dumps(msg).encode('utf-8'))
BUFFER = Queue.Queue()
PKTS = []
def capture():
    global BUFFER
    global PKTS
    ip = ni.ifaddresses(IFACE)[2][0]['addr']
    while(1):
        #Network to flowtbag format
        PKTS = []
        sniff(timeout=WINDOW, iface=IFACE, store=0, prn=noNSH)
        pkts = [(len(pkt), str(pkt), pkt.time) for pkt in PKTS]
        BUFFER.put(pkts)

def write_flows():
    global BUFFER
    global TOPIC
    global producer
    while (1):
        pkts = BUFFER.get()
        flows=flowtbag.Flowtbag(pkts)
        salvar_fluxosKafka(flows, CLASS, TOPIC, producer)

#Start simple threads
capture_thread = threading.Thread(target=capture)
writer_thread = threading.Thread(target=write_flows)

capture_thread.start()
writer_thread.start()
```

Envio de dados ao produtor do Kafka para serem consumidos pelo consumidor do Spark

COPPE UFRJ

CATF
Car

A / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

```
    try:
        PKTS.append(pkt)
    except:
        pass

#Send to Kafka
def salvar_fluxosKafka(flows, cl
    for flow in flows.ac
        msg = str(flo
        producer.se
BUFFER = Queue.Queue
PKTS = []
def capture():
    global BUFFER
    global PKTS
    ip = ni.ifaddresses(IFACE)[2][0]['addr']
    while(1):
        #Network to flowtbag format
        PKTS = []
        sniff(timeout=WINDOW, iface=IFACE, store=0, prn=noNSH)
        pkts = [(len(pkt), str(pkt), pkt.time) for pkt in PKTS]
        BUFFER.put(pkts)

def write_flows():
    global BUFFER
    global TOPIC
    global producer
    while (1):
        pkts = BUFFER.get()
        flows=flowtbag.Flowtbag(pkts)
        salvar_fluxosKafka(flows, CLASS, TOPIC, producer)

#Start simple threads
capture_thread = threading.Thread(target=capture)
writer_thread = threading.Thread(target=write_flows)

capture_thread.start()
writer_thread.start()
```

Função de caputra dos pacotes nas interface de rede e cópia dos pacotes para um *buffer* que é lido pelo Flowtbag

COPPE UFRJ

CATP
Car

A / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

```
    try:
        PKTS.append(pkt)
    except:
        pass

#Send to Kafka
def salvar_fluxosKafka(flows, classe, topico, producer):
    for flow in flows.active_flows.values():
        msg = str(flow) + ',' + classe
        producer.send(topico, json.dumps(msg).encode('utf-8'))
BUFFER = Queue.Queue()
PKTS = []
def capture():
    global BUFFER
    global PKTS
    ip = ni.ifaddresses(IFACE)[2][0]['addr']
    while(1):
        #Network to flowtbag form
        PKTS = []
        sniff(timeout=WINDO
        pkts = [(len(pkt),
        BUFFER.put(pkts)

def write_flows():
    global BUFFER
    global TOPIC
    global producer
    while (1):
        pkts = BUFFER.get()
        flows=flowtbag.Flowtbag(pkts)
        salvar_fluxosKafka(flows, CLASS, TOPIC, producer)

#Start simple threads
capture_thread = threading.Thread(target=capture)
writer_thread = threading.Thread(target=write_flows)

capture_thread.start()
writer_thread.start()
```

Recuperação dos fluxos gerados pelo flowtbag e escrita no produtor do Kafka

COPPE UFRJ

CATF
Car

A / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

```
    try:
        PKTS.append(pkt)
    except:
        pass

#Send to Kafka
def salvar_fluxosKafka(flows, classe, topico, producer):
    for flow in flows.active_flows.values():
        msg = str(flow) + ',' + classe
        producer.send(topico, json.dumps(msg).encode('utf-8'))
BUFFER = Queue.Queue()
PKTS = []
def capture():
    global BUFFER
    global PKTS
    ip = ni.ifaddresses(IFACE)[2][0]['addr']
    while(1):
        #Network to flowtbag format
        PKTS = []
        sniff(timeout=WINDOW, iface=IFACE, store=0, prn=noNSH)
        pkts = [(len(pkt), str(pkt), pkt.time) for pkt in PKTS]
        BUFFER.put(pkts)

def write_flows():
    global BUFFER
    global TOPIC
    global producer
    while (1):
        pkts = BUFFER.get()
        flows=flowtbag.Flowtbag(pkts)
        salvar_fluxosKafka(flows, CLASS, TOPIC, prod

#Start simple threads
capture_thread = threading.Thread(target=capture)
writer_thread = threading.Thread(target=writ

capture_thread.start()
writer_thread.start()
```

Execução paralela das funções de captura de pacotes e escrita no produtor do Kafka através de *threads*

UFRJ

CATR
Car

A / UFRJ
TELEINFORMÁTICA E AUTOMAÇÃO

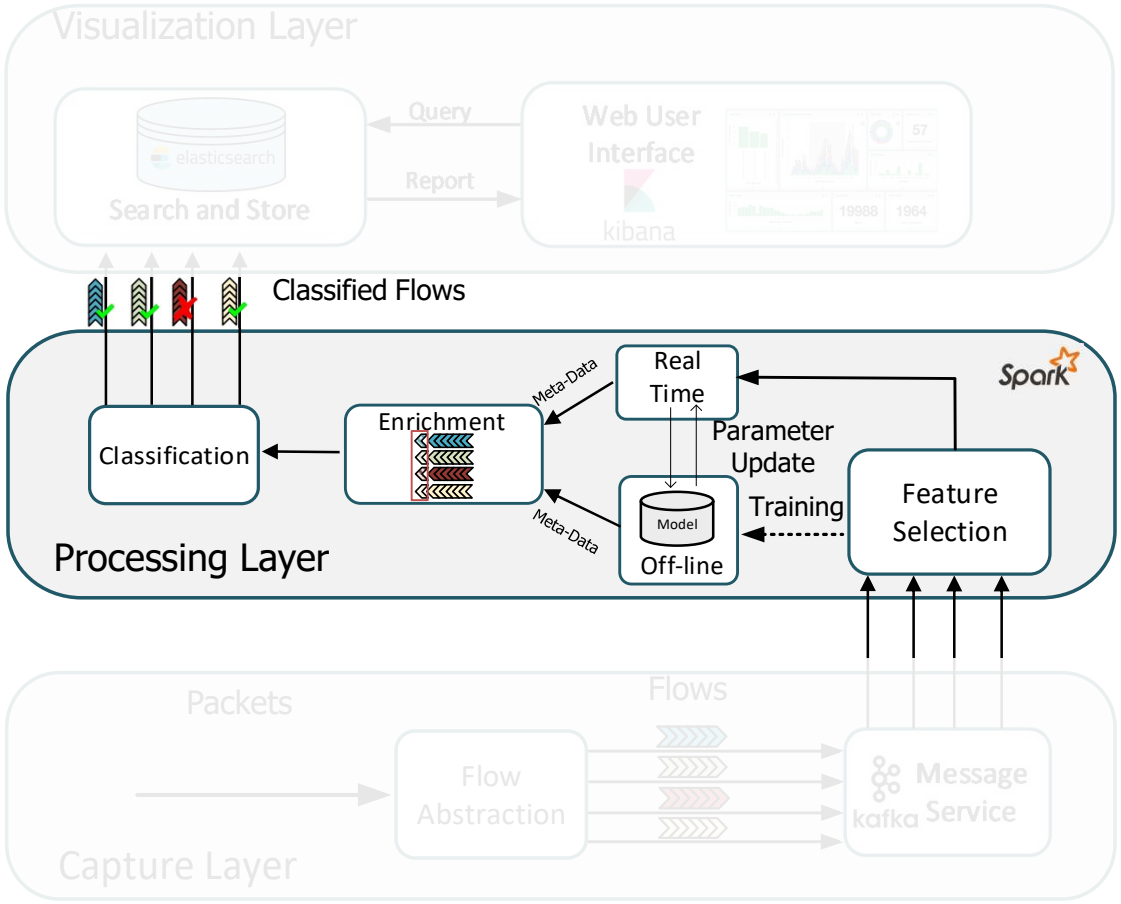
```
    try:
        PKTS.append(pkt)
    except:
        pass

#Send to Kafka
def salvar_fluxosKafka(flows, classe, topico, producer):
    for flow in flows.active_flows.values():
        msg = str(flow) + ',' + classe
        producer.send(topico, json.dumps(msg).encode('utf-8'))
BUFFER = Queue.Queue()
PKTS = []
def capture():
    global BUFFER
    global PKTS
    ip = ni.ifaddresses(IFACE)[2][0]['addr']
    while(1):
        #Network to flowtbag format
        PKTS = []
        sniff(timeout=WINDOW, iface=IFACE, store=0, prn=noNSH)
        pkts = [(len(pkt), str(pkt), pkt.time) for pkt in PKTS]
        BUFFER.put(pkts)

def write_flows():
    global BUFFER
    global TOPIC
    global producer
    while (1):
        pkts = BUFFER.get()
        flows=flowtbag.Flowtbag(pkts)
        salvar_fluxosKafka(flows, CLASS, TOPIC, producer)

#Start simple threads
                                                    ure)
                                                    _flows)
```

Código executa em máquinas que são monitoradas pelo CATRACA

COPPE UFRJ

CATRACA
Camada de Processamento

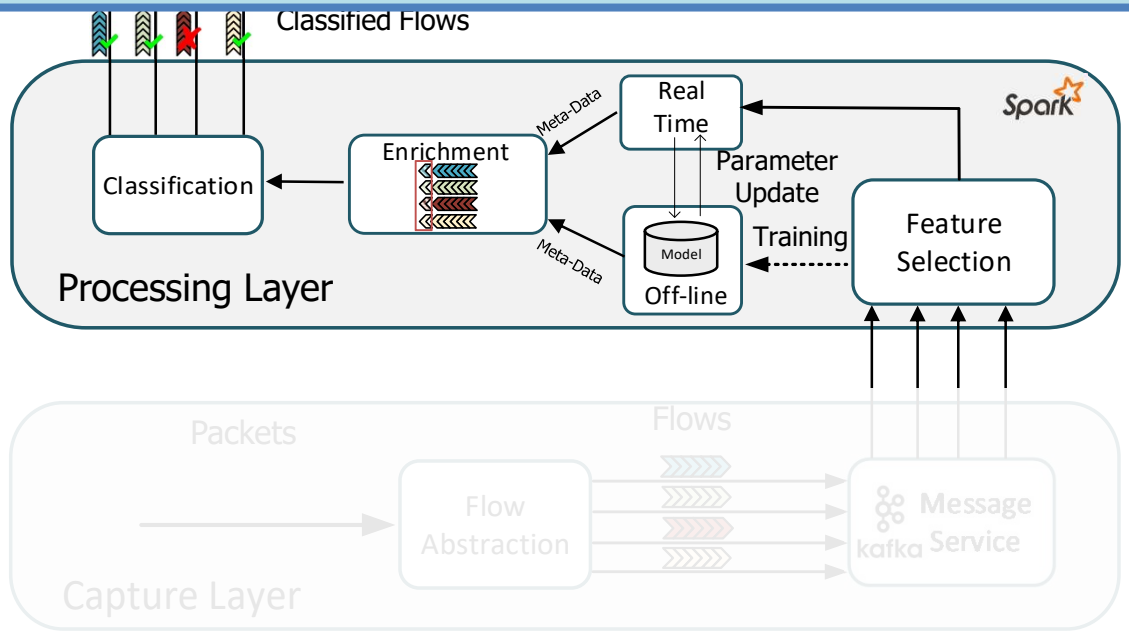
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Execução no *cluster* Spark

spark-submit --master spark://master:7077 --packages TargetHolding:pyspark-elastic:0.4.2 --jars /opt/spark/jars/elasticsearch-spark-20_2.10-5.5.1.jar, /opt/spark/jars/spark-streaming-kafka-0-8-assembly_2.11-2.1.1.jar --conf spark.executor.extraJavaOptions=" -XX:MaxPermSize=15G " <arquivo.py> hdfs://master:9000/user/app/<modelo treinado> 10.10.10.3:2181 topic1



COPPE UFRJ

```
from __future__ import print_function

import sys

from pyspark_elastic import EsSparkContext
from pyspark.mllib.linalg import Vectors
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils, OffsetRange
import json
import geoip2
import time

#### ML
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
from pyspark.mllib.util import MLUtils
####


####
import numpy as np
from pyspark.mllib.stat import Statistics
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.util import MLUtils
from tempfile import NamedTemporaryFile
####

from py4j.protocol import Py4JJavaError
import os
import subprocess

####for firewall
import httplib
import json



#numero de caracteristicas e classes do dataset
numberFeatures=46
numberClasses=2

#ip do firewall que vai atuar sobre os fluxos em tempo real
ipFirewall='10.240.114.45'

#funções de apoio
def convertTofloat(x):
        for i in range(len(x)):
                        x[i]=float(x[i])
        return x

def convertToString(x):
        for i in range(len(x)):
                x[i]=str(x[i])
        return x


def dataPreparing(lines):
        #os dados chegam como string e são convertidas para vetor de float
        virgulas  = lines.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures]))
        vectors = virgulas.mapValues(lambda x: np.array(x)) #converte os valores em arrays
        test = vectors.map(lambda x:x[1]) #pega so os valores
        classes = test.map(lambda x:x[numberFeatures-5]) #recuepra a classe
        test = test.map(lambda x:x[0:numberFeatures-5]) #remove a classe dos dados
```

CA
C

```
from __future__ import print_function

import sys

from pyspark_elastic import EsSparkContext
from pyspark.mllib.linalg import Vectors
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils, OffsetRange
import json
import geoip2
import time

#### ML
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
from pyspark.mllib.util import MLUtils
####


####
import numpy as np
from pyspark.mllib.stat import Statistics
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.util import MLUtils
from tempfile import NamedTemporaryFile
####

from py4j.protocol import Py4JJavaError
import os
import subprocess

####for firewall
import httplib
import json
```

Importação dos pacotes necessários Spark, numpy e mllib são os principais

```
#numero de caracteristicas e classes do dataset
numberFeatures=46
numberClasses=2

#ip do firewall que vai atuar sobre os fluxos em tempo real
ipFirewall='10.240.114.45'

#funções de apoio
def convertTofloat(x):
        for i in range(len(x)):
                        x[i]=float(x[i])
        return x

def convertToString(x):
        for i in range(len(x)):
                x[i]=str(x[i])
        return x


def dataPreparing(lines):
        #os dados chegam como string e são convertidas para vetor de float
        virgulas  = lines.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures]))
        vectors = virgulas.mapValues(lambda x: np.array(x)) #converte os valores em arrays
        test = vectors.map(lambda x:x[1]) #pega so os valores
        classes = test.map(lambda x:x[numberFeatures-5]) #recuepra a classe
        test = test.map(lambda x:x[0:numberFeatures-5]) #remove a classe dos dados
```

```
from __future__ import print_function

import sys

from pyspark_elastic import EsSparkContext
from pyspark.mllib.linalg import Vectors
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils, OffsetRange
import json
import geoip2
import time

#### ML
from pyspark.mllib.tree import DecisionTree, DecisionTreeModel
from pyspark.mllib.util import MLUtils
####


####
import numpy as np
from pyspark.mllib.stat import Statistics
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.util import MLUtils
from tempfile import NamedTemporaryFile
####

from py4j.protocol import Py4JJavaError
import os
import subprocess

####for firewall
import httplib
import json



#numero de caracteristicas e classes do dataset
numberFeatures=46
numberClasses=2

#ip do firewall que vai atuar sobre os fluxos em tempo real
ipFirewall='10.240.114.45'

#funções de apoio
def convertTofloat(x):
        for i in range(len(x)):
                        x[i]=float(x[i])
        return x

def convertToString(x):
        for i in range(len(x)):
                x[i]=str(x[i])
        return x


def dataPreparing(lines):
        #os dados chegam como string e são convertidas para vetor de float
        virgulas  = lines.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures]))
        vectors = virgulas.mapValues(lambda x: np.array(x)) #converte os valores em arrays
        test = vectors.map(lambda x:x[1]) #pega so os valores
        classes = test.map(lambda x:x[numberFeatures-5]) #recuepra a classe
        test = test.map(lambda x:x[0:numberFeatures-5]) #remove a classe dos dados
```

Funções de preparação dos dados
Conversão de strings em arrays para serem usados no MLLib

```
def CorrelationFeature(vectors):

        matriz=sc.broadcast(Statistics.corr(vectors, method="pearson"))

        summary = Statistics.colStats(vectors)

        varianza=summary.variance()


        #Calculo dos pesos para seleção de características
        w={}
        aij={}
        for i in range(len(matriz.value)):
                w[i]=0
                aij[i]=0
                for j in np.nan_to_num(matriz.value[i]):
                        k=abs(j)
                        aij[i]=aij[i]+k
                w[i]=varianza[i]/aij[i]

        #ordenamento dos índices de acordo com os pesos de cada característica
        r=sorted([(value,key) for (key,value) in w.items()],reverse=True)

        index=[]
        for i in r:
                index.append(i[1])

        index=index[0:6] #tacking the first 6 features

        #retorna os indices
        return index


def MatrixReducer(vectors, index):

        reducedMatrix =[]
        vectors = np.matrix(vectors)

        #seleciona somente as colunas relativas as características de maior peso
        #indice das características são passados por index
        for k in index:
                reducedMatrix.append(vectors[:,k]) #reduced matrix

        vectors2 = np.column_stack(reducedMatrix)
        vectors2 = np.array(vectors2)

        return vectors2


#A entrada dos dados no MLib tem que ser LibSVM
#conversão do array para libsvm
def pass2libsvm(vectors2,classes):

        newVector=classes.zip(vectors2)
        grouped=newVector.groupByKey().mapValues(list)
        final=newVector.map(lambda x : LabeledPoint(x[0],x[1]))


        return final
```

```
def CorrelationFeature(vectors):

        matriz=sc.broadcast(Statistics.corr(vectors, method="pearson"))

        summary = Statistics.colStats(vectors)

        varianza=summary.variance()


        #Calculo dos pesos para seleção de características
        w={}
        aij={}
        for i in range(len(matriz.value)):
                w[i]=0
                aij[i]=0
                for j in np.nan_to_num(matriz.value[i]):
                        k=abs(j)
                        aij[i]=aij[i]+k
                w[i]=varianza[i]/aij[i]

        #ordenamento dos índices de acordo com os pesos de cada característica
        r=sorted([(value,key) for (key,value) in w.items()],reverse=True)

        index=[]
        for i in r:
                index.append(i[1])

        index=index[0:6] #tacking the first 6 features

        #retorna os indices
        return index


def MatrixReducer(vectors, index):

        reducedMatrix =[]
        vectors = np.matrix(vectors)

        #seleciona somente as colunas relativas as características de maior peso
        #indice das características são passados por index
        for k in index:
                reducedMatrix.append(vectors[:,k]) #reduced matrix

        vectors2 = np.column_stack(reducedMatrix)
        vectors2 = np.array(vectors2)

        return vectors2


#A entrada dos dados no MLib tem que ser LibSVM
#conversão do array para libsvm
def pass2libsvm(vectors2,classes):

        newVector=classes.zip(vectors2)
        grouped=newVector.groupByKey().mapValues(list)
        final=newVector.map(lambda x : LabeledPoint(x[0],x[1]))


        return final
```

Calculo dos índices das Características com maior Relevância para definição Da classificação

```
def CorrelationFeature(vectors):

        matriz=sc.broadcast(Statistics.corr(vectors, method="pearson"))

        summary = Statistics.colStats(vectors)

        varianza=summary.variance()


        #Calculo dos pesos para seleção de características
        w={}
        aij={}
        for i in range(len(matriz.value)):
                w[i]=0
                aij[i]=0
                for j in np.nan_to_num(matriz.value[i]):
                        k=abs(j)
                        aij[i]=aij[i]+k
                w[i]=varianza[i]/aij[i]

        #ordenamento dos índices de acordo com os pesos de cada característica
        r=sorted([(value,key) for (key,value) in w.items()],reverse=True)

        index=[]
        for i in r:
                index.append(i[1])

        index=index[0:6] #tacking the first 6 features

        #retorna os indices
        return index


def MatrixReducer(vectors, index):

        reducedMatrix =[]
        vectors = np.matrix(vectors)

        #seleciona somente as colunas relativas as características de maior peso
        #indice das características são passados por index
        for k in index:
                reducedMatrix.append(vectors[:,k]) #reduced matrix

        vectors2 = np.column_stack(reducedMatrix)
        vectors2 = np.array(vectors2)

        return vectors2


#A entrada dos dados no MLib tem que ser LibSVM
#conversão do array para libsvm
def pass2libsvm(vectors2,classes):

        newVector=classes.zip(vectors2)
        grouped=newVector.groupByKey().mapValues(list)
        final=newVector.map(lambda x : LabeledPoint(x[0],x[1]))


        return final
```

Redução da matriz de Características para aquelas que têm apenas os índices em index

```
def CorrelationFeature(vectors):

        matriz=sc.broadcast(Statistics.corr(vectors, method="pearson"))

        summary = Statistics.colStats(vectors)

        varianza=summary.variance()


        #Calculo dos pesos para seleção de características
        w={}
        aij={}
        for i in range(len(matriz.value)):
                w[i]=0
                aij[i]=0
                for j in np.nan_to_num(matriz.value[i]):
                        k=abs(j)
                        aij[i]=aij[i]+k
                w[i]=varianza[i]/aij[i]

        #ordenamento dos índices de acordo com os pesos de cada característica
        r=sorted([(value,key) for (key,value) in w.items()],reverse=True)

        index=[]
        for i in r:
                index.append(i[1])

        index=index[0:6] #tacking the first 6 features

        #retorna os indices
        return index


def MatrixReducer(vectors, index):

        reducedMatrix =[]
        vectors = np.matrix(vectors)
                                                                    de maior peso

        return


#A entrada d    os no MLib tem que ser LibSVM
#conversão d  array para libsvm
def pass2libsvm(vectors2,classes):

        newVector=classes.zip(vectors2)
        grouped=newVector.groupByKey().mapValues(list)
        final=newVector.map(lambda x : LabeledPoint(x[0],x[1]))


        return final
```

Conversão dos dados para o formato da libSVM usado pelo MLLib

CA
C

```
#envio de regras de bloqueio para o firewall
def blockFlows(flow):
        vec = flow[0]
        prediction = flow[1]
        #prediction.map(lambda x: x).pprint()
        if prediction != 0.0:
                tupla = json.loads(vec)
                ipSrc=tupla[0]
                ipDst=tupla[2]
                conn = httplib.HTTPConnection(ipFirewall,8000)
                #firewall restfull - Envio
                conn.request("POST","/add",json.dumps({'ipSrc':ipSrc, 'ipDst':ipDst}))
                res=conn.getresponse()
                res.read()
        return flow


#funcao de verificacao se o arquivo existe no HDFS
def path_exist(file):
        path='hdfs://master:9000/user/app/'
        try:
                cmd = ['hdfs', 'dfs', '-find',path]
                files = subprocess.check_output(cmd).strip().split('\n')
                if file in files:
                        return True
                else:
                        return False
        except Py4JJavaError as e:
                return False


#Recupera o modelo de ML já calculado
#ou calcula um novo modelo
def getModel(path,file):

        if path_exist(path+'index-'+file):
                index=sc.textFile(path+'index-'+file)
                a=index.collect()
                b=lambda x : [ int(i) for i in x ]

                return DecisionTreeModel.load(sc, path+'model-'+file), b(a)


        else:

                vector,classes = dataPreparing(sc.textFile(path+file))

                index=CorrelationFeature(vector) #se precisar de feature do Feature Selection

                reduced=MatrixReducer(vector,index)

                data=pass2libsvm(reduced,classes)


                model = DecisionTree.trainClassifier(data, numberClasses,{})

                model.save(sc, path+'model-'+file)

                return  model, index
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
```

CA
C

```
#envio de regras de bloqueio para o firewall
def blockFlows(flow):
        vec = flow[0]
        prediction = flow[1]
        #prediction.map(lambda x: x).p
        if prediction != 0.0:
                tupla = json.loads(vec)
                ipSrc=tupla[0]
                ipDst=tupla[2]
                conn = httplib.HTTPConnection(ipFirewall,800
                #firewall restfull - Envio
                conn.request("POST","/add",json.dumps({'ipSrc':i
                res=conn.getresponse()
                res.read()
        return flow


#funcao de verificacao se o arquivo existe no HDFS
def path_exist(file):
        path='hdfs://master:9000/user/app/'
        try:
                cmd = ['hdfs', 'dfs', '-find',path]
                files = subprocess.check_output(cmd).strip().split('\n')
                if file in files:
                        return True
                else:
                        return False
        except Py4JJavaError as e:
                return False


#Recupera o modelo de ML já calculado
#ou calcula um novo modelo
def getModel(path,file):

        if path_exist(path+'index-'+file):
                index=sc.textFile(path+'index-'+file)
                a=index.collect()
                b=lambda x : [ int(i) for i in x ]

                return DecisionTreeModel.load(sc, path+'model-'+file), b(a)


        else:

                vector,classes = dataPreparing(sc.textFile(path+file))

                index=CorrelationFeature(vector) #se precisar de feature do Feature Selection

                reduced=MatrixReducer(vector,index)

                data=pass2libsvm(reduced,classes)


                model = DecisionTree.trainClassifier(data, numberClasses,{})

                model.save(sc, path+'model-'+file)

                return  model, index
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
```

Ao detectar um fluxo malicioso, realiza uma chamada Web para o FW para bloquear o fluxo

FRJ
AUTOMAÇÃO

CA
C

```
#envio de regras de bloqueio para o firewall
def blockFlows(flow):
        vec = flow[0]
        prediction = flow[1]
        #prediction.map(lambda x: x).pprint()
        if prediction != 0.0:
                tupla = json.loads(vec)
                ipSrc=tupla[0]
                ipDst=tupla[2]
                conn = httplib.HTTPConnection(ipFirewall,8000)
                #firewall restfull - Envio
                conn.request("POST","/add",json.dumps({'ipSrc':ipSrc,
                res=conn.getresponse()
                res.read()
        return flow


#funcao de verificacao se o arquivo existe no HDFS
def path_exist(file):
        path='hdfs://master:9000/user/app/'
        try:
                cmd = ['hdfs', 'dfs', '-find',path]
                files = subprocess.check_output(cmd).strip().split('\n')
                if file in files:
                        return True
                else:
                        return False
        except Py4JJavaError as e:
                return False


#Recupera o modelo de ML já calculado
#ou calcula um novo modelo
def getModel(path,file):

        if path_exist(path+'index-'+file):
                index=sc.textFile(path+'index-'+file)
                a=index.collect()
                b=lambda x : [ int(i) for i in x ]

                return DecisionTreeModel.load(sc, path+'model-'+file), b(a)


        else:

                vector,classes = dataPreparing(sc.textFile(path+file))

                index=CorrelationFeature(vector) #se precisar de feature do Feature Selection

                reduced=MatrixReducer(vector,index)

                data=pass2libsvm(reduced,classes)


                model = DecisionTree.trainClassifier(data, numberClasses,{})

                model.save(sc, path+'model-'+file)

                return  model, index
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
```

Verificação de se um determinado arquivo existe no HDFS

RJ
AUTOMAÇÃO

CA
C

```
#envio de regras de bloqueio para o firewall
def blockFlows(flow):
        vec = flow[0]
        prediction = flow[1]
        #prediction.map(lambda x: x).pprint()
        if prediction != 0.0:
                tupla = json.loads(vec)
                ipSrc=tupla[0]
                ipDst=tupla[2]
                conn = httplib.HTTPConnection(ipFirewall,8000)
                #firewall restfull - Envio
                conn.request("POST","/add",json.dumps({'ipSrc':ipSrc, 'ip
                res=conn.getresponse()
                res.read()
        return flow


#funcao de verificacao se o arquivo existe no HDFS
def path_exist(file):
        path='hdfs://master:9000/user/app/'
        try:
                cmd = ['hdfs', 'dfs', '-find',path]
                files = subprocess.check_output(cmd).strip
                if file in files:
                        return True
                else:
                        return False
        except Py4JJavaError as e:
                return False


#Recupera o modelo de ML já cal
#ou calcula um novo modelo
def getModel(path,file):

        if path_exist(path+'index-'+file):
                index=sc.textFile(path+'index-'+file)
                a=index.collect()
                b=lambda x : [ int(i) for i in x ]

                return DecisionTreeModel.load(sc, path+'model-'+file), b(a)


        else:

                vector,classes = dataPreparing(sc.textFile(path+file))

                index=CorrelationFeature(vector) #se precisar de feature do Feature Selection

                reduced=MatrixReducer(vector,index)

                data=pass2libsvm(reduced,classes)


                model = DecisionTree.trainClassifier(data, numberClasses,{})

                model.save(sc, path+'model-'+file)

                return  model, index
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
```

Recuperação do modelo treinado a partir do HDFS
Caso não exista, treina um novo modelo com dados *offline*

CA-
C

```
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
        locSrcIp = geoip2.geolite2.lookup(dictX['srcip'])
        locDstIp = geoip2.geolite2.lookup(dictX['dstip'])

        try:

                if locSrcIp and locSrcIp.location:

                        dictX['srclocation']= {'lat': locSrcIp.location[0], 'lon':locSrcIp.location[1]}
                else:
                        dictX['srclocation']= {'lat': 48, 'lon':22}


                if locDstIp and locDstIp.location:
                        dictX['dstlocation']= {'lat': locSrcIp.location[0], 'lon':locSrcIp.location[1]}
                else:
                        dictX['dstlocation']= {'lat': 48, 'lon':22}
        except AttributeError:

                pass
        except TypeError:
                pass

        return dictX


#Execução do main
if __name__ == "__main__":
        if len(sys.argv) != 4:
                exit(-1)

        sc = SparkContext(appName="CATRACA with DT")

        #Create model
        a=0
        orig=sys.argv[1]
        path='hdfs://master:9000/user/app/'
        file=orig.split('app/')[1]
        features=sc.textFile(path+'feature-des.txt').collect()
        feat=[]
        for i in features:
                feat.append(i.split('-')[0].split(' ')[0])


        #se o modelo ainda não existe no HDFS, calcula-se um modelo com dados offline
        [model,index]=getModel(path,file)

        if path_exist(path+'index-'+file) == False:
                rdd=sc.parallelize(index)
                rdd.saveAsTextFile(path+'index-'+file)

        ####Streaming

        ssc = StreamingContext(sc, 1)


        ###kafka
        zkQuorum, topic = sys.argv[2:]
```

CA-
C

```
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
        locSrcIp = geoip2.geolite2.lookup(dictX['srcip'])
        locDstIp = geoip2.geolite2.lookup(dictX['dstip'])

        try:

                if locSrcIp and locSrcIp.location:

                        dictX['srclocation']= {'lat': locSrcIp.location[0], 'lon':locSrcIp.location[1]}
                else:
                        dictX['srclocation']= {'lat': 48, 'lon':22}


                if locDstIp and locDstIp.location:
                        dictX['dstlocation']= {'lat': locSrcIp.location[0], 'lon':locSrcIp.location[1]}
                else:
                        dictX['dstlocation']= {'lat': 48, 'lon':22}
        except AttributeError:

                pass
        except TypeError:
                pass

        return dictX


#Execução do main
if __name__ == "__main__":
        if len(sys.argv) != 4:
                exit(-1)

        sc = SparkContext(appName="CATRACA with DT")

        #Create model
        a=0
        orig=sys.argv[1]
        path='hdfs://master:9000/user/app/'
        file=orig.split('app/')[1]
        features=sc.textFile(path+'feature-des.txt').collect()
        feat=[]
        for i in features:
                feat.append(i.split('-')[0].split(' ')[0])


        #se o modelo ainda não existe no HDFS, calcula-se um modelo com dados offline
        [model,index]=getModel(path,file)

        if path_exist(path+'index-'+file) == False:
                rdd=sc.parallelize(index)
                rdd.saveAsTextFile(path+'index-'+file)

        ####Streaming

        ssc = StreamingContext(sc, 1)


        ###kafka
        zkQuorum, topic = sys.argv[2:]
```

Enriquecimento com a localização baseada no IP

CA-
C

```
#Adiciona a localização como característica do conjunto de dados
#localização obtida pelo IP
def addLocation (x):
        dictX = dict(x)
        locSrcIp = geoip2.geolite2.lookup(dictX['srcip'])
        locDstIp = geoip2.geolite2.lookup(dictX['dstip'])

        try:

                if locSrcIp and locSrcIp.location:

                        dictX['srclocation']= {'lat': locSrcIp.location[0], 'lon':locSrcIp.location[1]}
                else:
                        dictX['srclocation']= {'lat': 48, 'lon':22}


                if locDstIp and locDstIp.location:
                        dictX['dstlocation']= {'lat': locSrcIp.location[0], 'lon':locSrcIp.location[1]}
                else:
                        dictX['dstlocation']= {'lat': 48, 'lon':22}
        except AttributeError:

                pass
        except TypeError:
                pass

        return dictX


#Execução do main
if __name__ == "__main__":
        if len(sys.argv) != 4:
                exit(-1)

        sc = SparkContext(appName="CATRACA with DT")

        #Create model
        a=0
        orig=sys.argv[1]
        path='hdfs://master:9000/user/app/'
        file=orig.split('app/')[1]
        features=sc.textFile(path+'feature-des.txt').collect()
        feat=[]
        for i in features:
                feat.append(i.split('-')[0].split(' ')[0])


        #se o modelo ainda não existe no HDFS, calcula-se um modelo com dados offline
        [model,index]=getModel(path,file)

        if path_exist(path+'index-'+file) == False:
                rdd=sc.parallelize(index)
                rdd.saveAsTextFile(path+'index-'+file)

        ####Streaming

        ssc = StreamingContext(sc, 1)


        ###kafka
        zkQuorum, topic = sys.argv[2:]
```

Carregamento do modelo de ML A partir do HDFS

```
###kafka
zkQuorum, topic = sys.argv[2:]

kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer", {topic: 1})

parsed = kvs.map(lambda v: json.loads(v[1]))

#recuperacao dos dados que chegam pelo kafka
lines  = parsed.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures-1])).mapValues(lambda x: convertTofloat(x))

#formatacao para o elastic
elastic=parsed.map(lambda x: x.split(',')).map(lambda x: {feat[i]: x[i] for i in range(numberFeatures-2)}).map(addLocation)

test = lines.flatMapValues(lambda x: MatrixReducer(x,index))

conf = {"es.resource" : "spark/test", "es.nodes" : "10.10.10.15", "es.index.auto.create": "true"}

vec = test.mapValues( Vectors.dense) #now we have the vectors with the format of the ML


try:
        vec=test.map(lambda x: x[1])
        #indexa os dados
        ips=test.transform(lambda x: x.keys().zipWithIndex()).map(lambda x: (x[1],x[0]))
        #aplica a predição nos dados
        algo=test.transform(lambda x: model.predict(x.values()).zipWithIndex()).map(lambda x: (x[1],x[0]))

        #junta os labels com os dados preditos
        joined = ips.join(algo).transform(lambda x: x.values())
        #força a execução do conjunto de transformações
        joined.foreachRDD(lambda v: print(v.collect()))


        #junta os dados que vão para o elastic com a predição
        yyy=elastic.transform(lambda x: x.zipWithIndex()).map(lambda x: (x[1],x[0]))
        toElastic = yyy.join(algo).transform(lambda x: x.values())

        almostSend=toElastic.map(lambda x: dict([i for i in x[0].items()+[('predict',x[1]),('timestamp',int(time.time()*1000))]]))

        now=almostSend.map(lambda x: ('key',x))

        ##Combina todos os dados dos RDDs sendo processados no momento e formata para ser salvo em HDFS
        ## Troca da classe de armazenamento para o ElasticSearch e, então, envio para o Elastic
        now.foreachRDD(lambda v: print(v.collect()))
        now.foreachRDD(lambda x: x.saveAsNewAPIHadoopFile(path='-',
                outputFormatClass="org.elasticsearch.hadoop.mr.EsOutputFormat",
                keyClass="org.apache.hadoop.io.NullWritable",
                valueClass="org.elasticsearch.hadoop.mr.LinkedMapWritable",conf=conf))
except AttributeError:
        pass

#execução do spark
ssc.start()
ssc.awaitTermination()
```

CA

C

```
###kafka
zkQuorum, topic = sys.argv[2:]

kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer", {topic: 1})

parsed = kvs.map(lambda v: json.loads(v[1]))

#recuperacao dos dados que chegam pelo kafka
lines  = parsed.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures-1])).mapValues(lambda x: convertTofloat(x))

#formatacao para o elastic
elastic=parsed.map(lambda x: x.split(',')).map(lambda x: {feat[i]: x[i] for i in range(numberFeatures-2)}).map(addLocation)

test = lines.flatMapValues(lambda x: MatrixReducer(x,index))

conf = {"es.resource" : "spark/test", "es.nodes" : "10.10.10.15", "es.index.auto.create": "true"}

vec = test.mapValues( Vectors.dense) #now we have the vectors with the format of the ML
```

Inicialização do Kafka e do Elastic

```
: x[1])

mbda x: x.keys().zipWithIndex()).map(lambda x: (x[1],x[0]))
s dados
ambda x: model.predict(x.values()).zipWithIndex()).map(lambda x: (x[1],x[0]))

os dados preditos
go).transform(lambda x: x.values())
conjunto de transformações
joined.foreachRDD(lambda v: print(v.collect()))


#junta os dados que vão para o elastic com a predição
yyy=elastic.transform(lambda x: x.zipWithIndex()).map(lambda x: (x[1],x[0]))
toElastic = yyy.join(algo).transform(lambda x: x.values())

almostSend=toElastic.map(lambda x: dict([i for i in x[0].items()+[('predict',x[1]),('timestamp',int(time.time()*1000))]]))

now=almostSend.map(lambda x: ('key',x))

##Combina todos os dados dos RDDs sendo processados no momento e formata para ser salvo em HDFS
## Troca da classe de armazenamento para o ElasticSearch e, então, envio para o Elastic
now.foreachRDD(lambda v: print(v.collect()))
now.foreachRDD(lambda x: x.saveAsNewAPIHadoopFile(path='-',
        outputFormatClass="org.elasticsearch.hadoop.mr.EsOutputFormat",
        keyClass="org.apache.hadoop.io.NullWritable",
        valueClass="org.elasticsearch.hadoop.mr.LinkedMapWritable",conf=conf))
except AttributeError:
        pass

#execução do spark
ssc.start()
ssc.awaitTermination()
```

CA
C

'RJ
JTOMAÇÃO

```
###kafka
zkQuorum, topic = sys.argv[2:]

kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer", {topic: 1})

parsed = kvs.map(lambda v: json.loads(v[1]))

#recuperacao dos dados que chegam pelo kafka
lines  = parsed.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures-1])).mapValues(lambda x: convertTofloat(x))

#formatacao para o elastic
elastic=parsed.map(lambda x: x.split(',')).map(lambda x: {feat[i]

test = lines.flatMapValues(lambda x: MatrixReducer(

conf = {"es.resource" : "spark/test", "es.no

vec = test.mapValues( Vectors.dense) #now w
```

Transformações nos dados e predição baseado no modelo carregado

```
try:
        vec=test.map(lambda x: x[1])
        #indexa os dados
        ips=test.transform(lambda x: x.keys().zipWithIndex()).map(lambda x: (x[1],x[0]))
        #aplica a predição nos dados
        algo=test.transform(lambda x: model.predict(x.values()).zipWithIndex()).map(lambda x: (x[1],x[0]))

        #junta os labels com os dados preditos
        joined = ips.join(algo).transform(lambda x: x.values())
        #força a execução do conjunto de transformações
        joined.foreachRDD(lambda v: print(v.collect()))


        #junta os dados que vão para o elastic com a predição
        yyy=elastic.transform(lambda x: x.zipWithIndex()).map(lambda x: (x[1],x[0]))
        toElastic = yyy.join(algo).transform(lambda x: x.values())

        almostSend=toElastic.map(lambda x: dict([i for i in x[0].items()+[('predict',x[1]),('timestamp',int(time.time()*1000))]]))

        now=almostSend.map(lambda x: ('key',x))

        ##Combina todos os dados dos RDDs sendo processados no momento e formata para ser salvo em HDFS
        ## Troca da classe de armazenamento para o ElasticSearch e, então, envio para o Elastic
        now.foreachRDD(lambda v: print(v.collect()))
        now.foreachRDD(lambda x: x.saveAsNewAPIHadoopFile(path='-',
                outputFormatClass="org.elasticsearch.hadoop.mr.EsOutputFormat",
                keyClass="org.apache.hadoop.io.NullWritable",
                valueClass="org.elasticsearch.hadoop.mr.LinkedMapWritable",conf=conf))
except AttributeError:
        pass

#execução do spark
ssc.start()
ssc.awaitTermination()
```


COPPE UFRJ

CA
C

RJ
JTOMAÇÃO

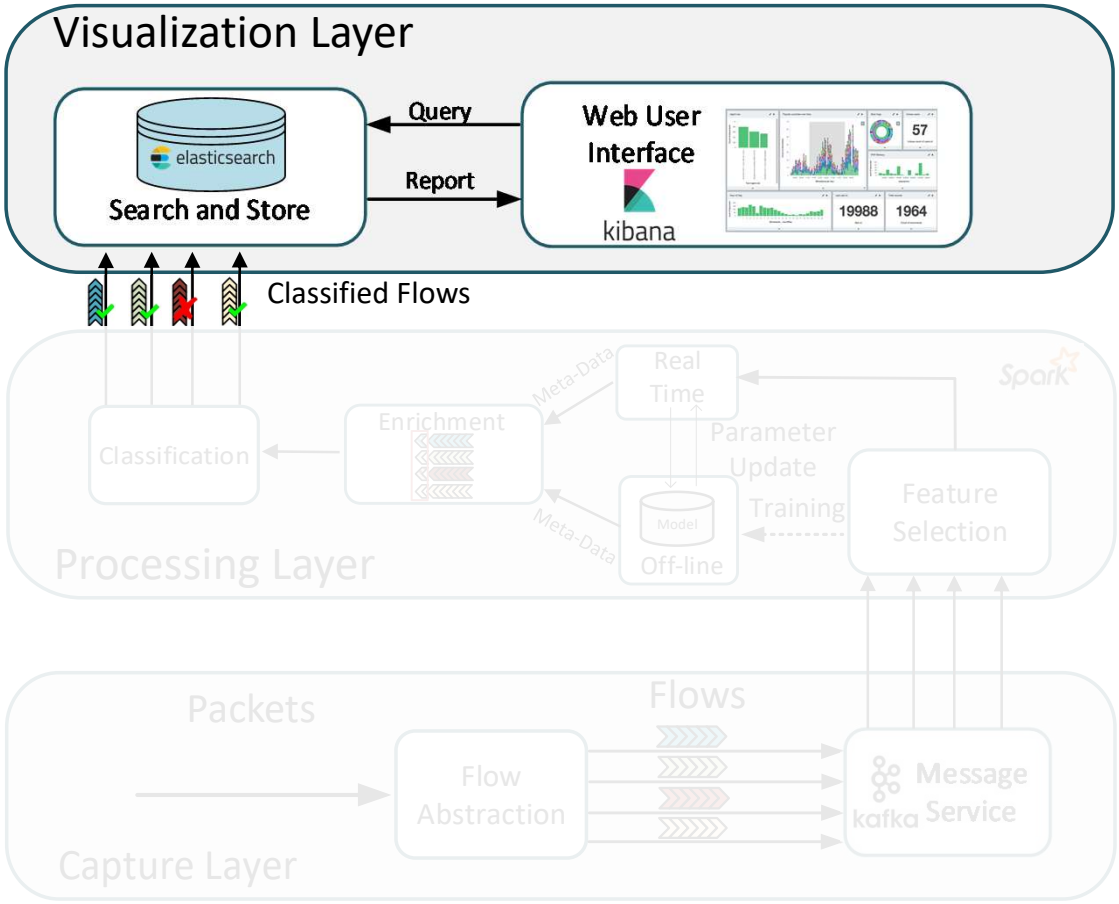
```
###kafka
zkQuorum, topic = sys.argv[2:]

kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer", {topic: 1})

parsed = kvs.map(lambda v: json.loads(v[1]))

#recuperacao dos dados que chegam pelo kafka
lines  = parsed.map(lambda x: x.split(',')).map(lambda x:(json.dumps(x[0:4]), x[4:numberFeatures-1])).mapValues(lambda x: convertTofloat(x))

#formatacao para o elastic
elastic=parsed.map(lambda x: x.split(',')).map(lambda x: {feat[i]: x[i] for i in range(numberFeatures-2)}).map(addLocation)

test = lines.flatMapValues(lambda x: MatrixReducer(x,index))

conf = {"es.resource" : "spark/test", "es.nodes" : "10.10.10.15", "es.index.auto.create": "true"}

vec = test.mapValues( Vectors.dense) #now we have the vectors with the format of the ML


try:
        vec=test.map(lambda x: x[1])
        #indexa os dados
        ips=test.transform(lambda x: x.keys().zipWithIndex()).map(lambda x: (x[1],x[0]))
        #aplica a predição nos dados
        algo=test.transform(lambda x: model.predict(x.values()).zipWithIndex()).map(lambda x: (x[1],x[0]))

        #junta os labels com os dados preditos
        joined = ips.join(algo).transform(lambda x: x.values())
        #força a execução do
        joined.foreachRD

        #junt
        yyy
        toE

        almos                                                                    ]),('timestamp',int(time.time()*1000))]]))

        now=almost

        ##Combi                                                                  ento e formata para ser salvo em HDFS
        ## Tr                                                                    a classe de armazenamento para o ElasticSearch e, então, envio para o Elastic
        now.foreachRDD(lambda v: print(v.collect()))
        now.foreachRDD(lambda x: x.saveAsNewAPIHadoopFile(path='-',
                outputFormatClass="org.elasticsearch.hadoop.mr.EsOutputFormat",
                keyClass="org.apache.hadoop.io.NullWritable",
                valueClass="org.elasticsearch.hadoop.mr.LinkedMapWritable",conf=conf))
except AttributeError:
        pass

#execução do spark
ssc.start()
ssc.awaitTermination()
```

Formatação e carregamento no ElasticSearch/Kibana

COPPE UFRJ

Arquitetura do Sistema CATRACA → Mais Detalhes

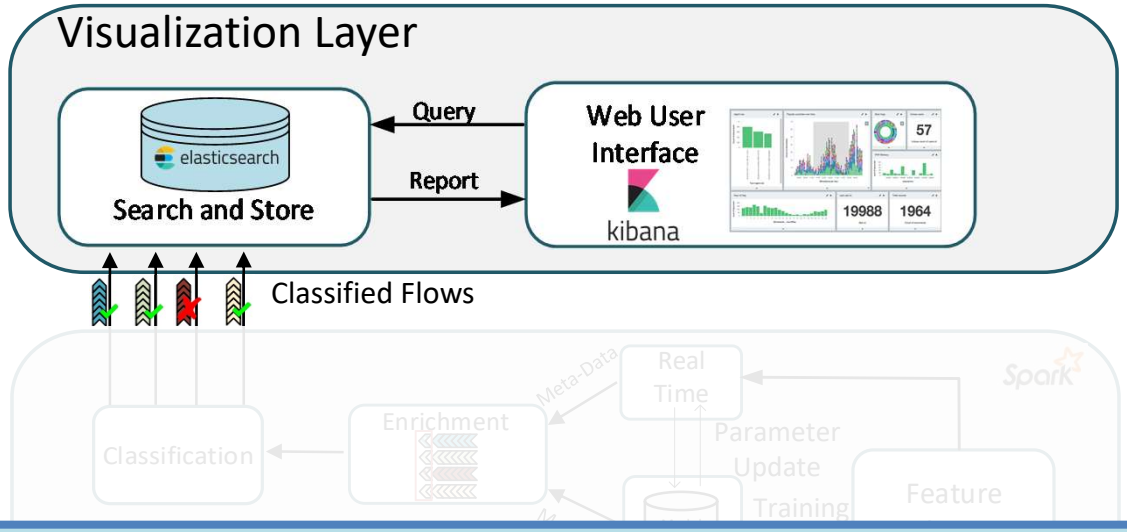
GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



COPPE UFRJ

Arquitetura do Sistema CATRACA → Mais Detalhes

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



Execução do Ambiente ElasticSearch

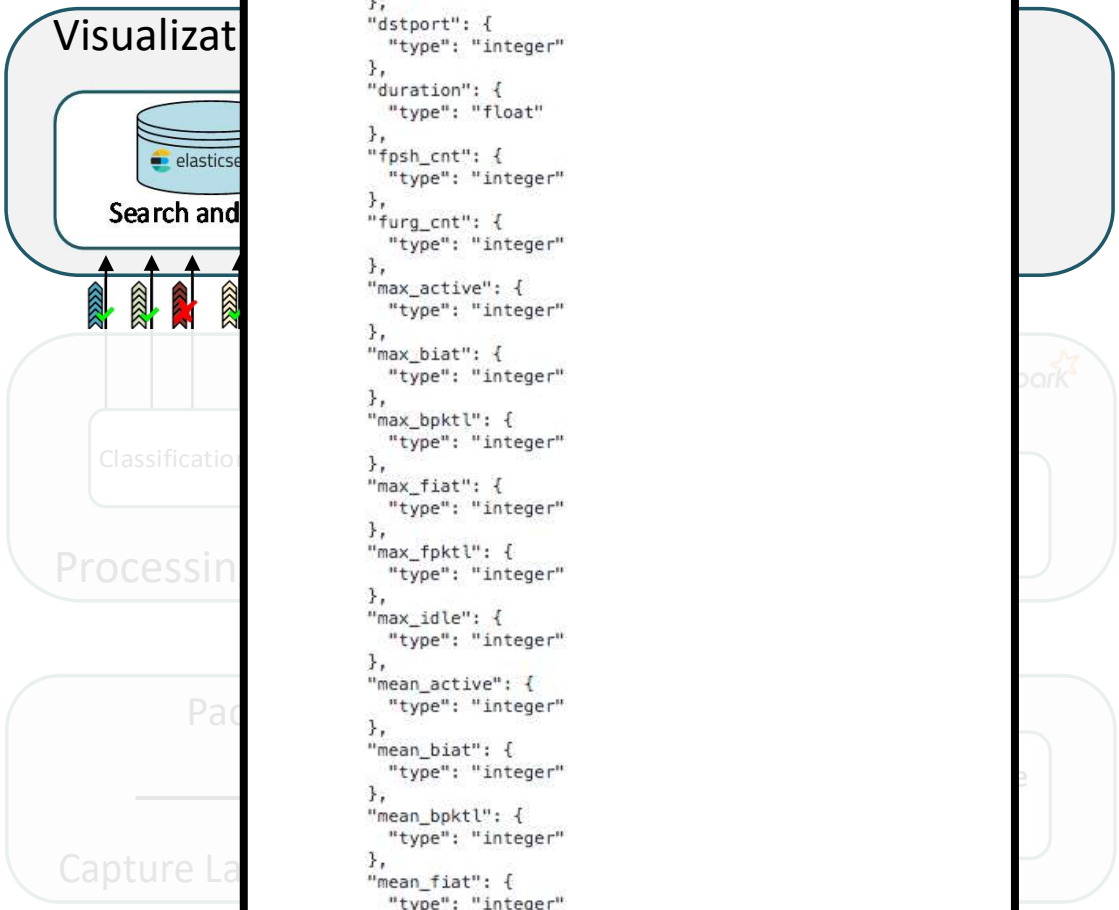
Os dados gerados pelo Spark são publicados no ElasticSearch através de chamadas HTTP

O modelo dos dados é representado por *mappings* → mapeamento do que o ElasticSearch recebe e tipos de dados pré-definidos

COPPE UFRJ

Arquitetura CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO




```
PUT spark
{
    "mappings": {
      "test": {
        "properties": {
          "bpsh_cnt": {
            "type": "integer"
          },
          "burg_cnt": {
            "type": "integer"
          },
          "dstip": {
            "type": "ip"
          },
          "dstlocation": {
            "type": "geo_point"
          },
          "dstport": {
            "type": "integer"
          },
          "duration": {
            "type": "float"
          },
          "fpsh_cnt": {
            "type": "integer"
          },
          "furg_cnt": {
            "type": "integer"
          },
          "max_active": {
            "type": "integer"
          },
          "max_biat": {
            "type": "integer"
          },
          "max_bpktl": {
            "type": "integer"
          },
          "max_fiat": {
            "type": "integer"
          },
          "max_fpktl": {
            "type": "integer"
          },
          "max_idle": {
            "type": "integer"
          },
          "mean_active": {
            "type": "integer"
          },
          "mean_biat": {
            "type": "integer"
          },
          "mean_bpktl": {
            "type": "integer"
          },
          "mean_fiat": {
            "type": "integer"
          },
          "mean_fpktl": {
            "type": "integer"
          },
          "mean_idle": {
            "type": "integer"
          },
```

COPPE UFRJ

```
"max_fiat": {
  "type": "integer"
},
"max_fpktl": {
  "type": "integer"
},
"max_idle": {
  "type": "integer"
},
"mean_active": {
  "type": "integer"
},
"mean_biat": {
  "type": "integer"
},
"mean_bpktl": {
  "type": "integer"
},
"mean_fiat": {
  "type": "integer"
},
"mean_fpktl": {
  "type": "integer"
},
"mean_idle": {
  "type": "integer"
},
"min_active": {
  "type": "integer"
},
"min_biat": {
  "type": "integer"
},
```

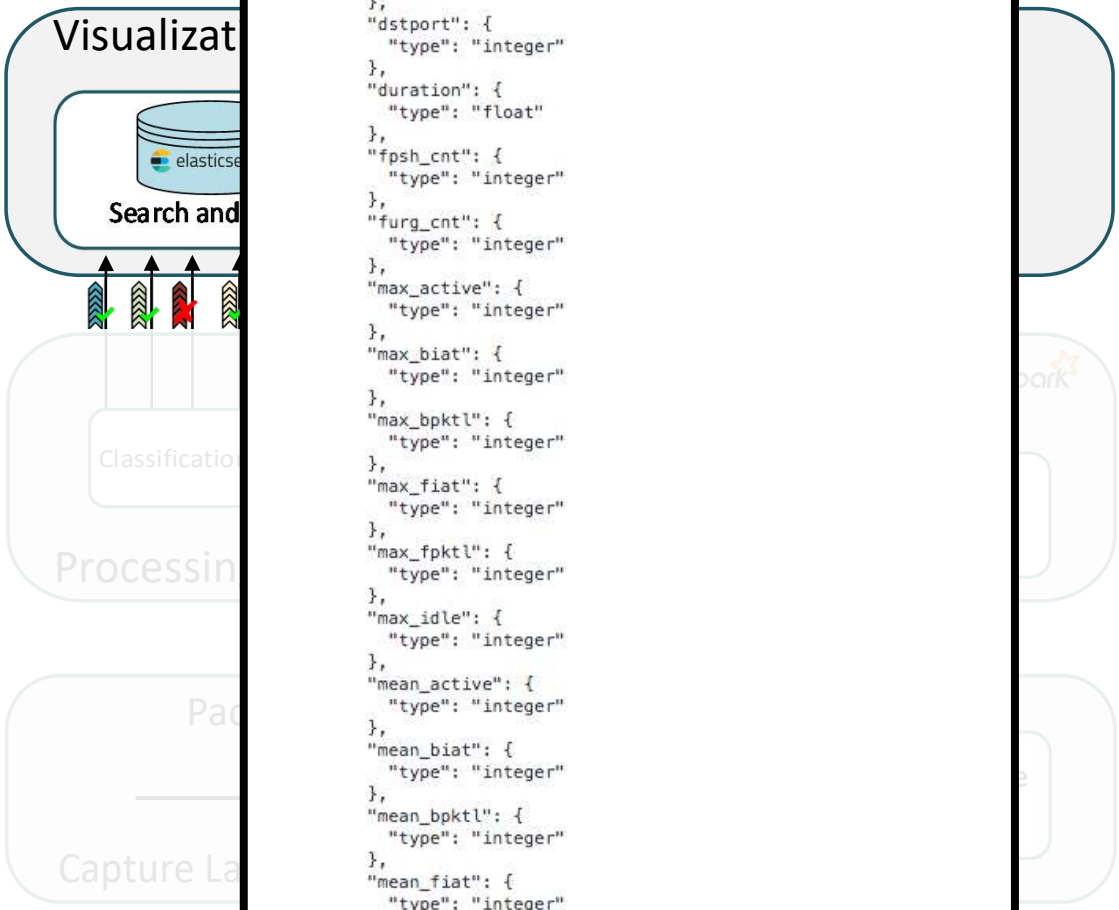
```
"max_fiat": {
  "type": "integer"
},
"max_fpktl": {
  "type": "integer"
},
"max_idle": {
  "type": "integer"
},
"mean_active": {
  "type": "integer"
},
"mean_biat": {
  "type": "integer"
},
"mean_bpktl": {
  "type": "integer"
},
"mean_fiat": {
  "type": "integer"
},
"mean_fpktl": {
  "type": "integer"
},
"mean_idle": {
  "type": "integer"
},
"min_active": {
  "type": "integer"
},
"min_biat": {
  "type": "integer"
},
```

Sintaxe do Mapeamento

"<nome-da-característica>":{
"type":"<tipo>"
},

Arquitetura CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

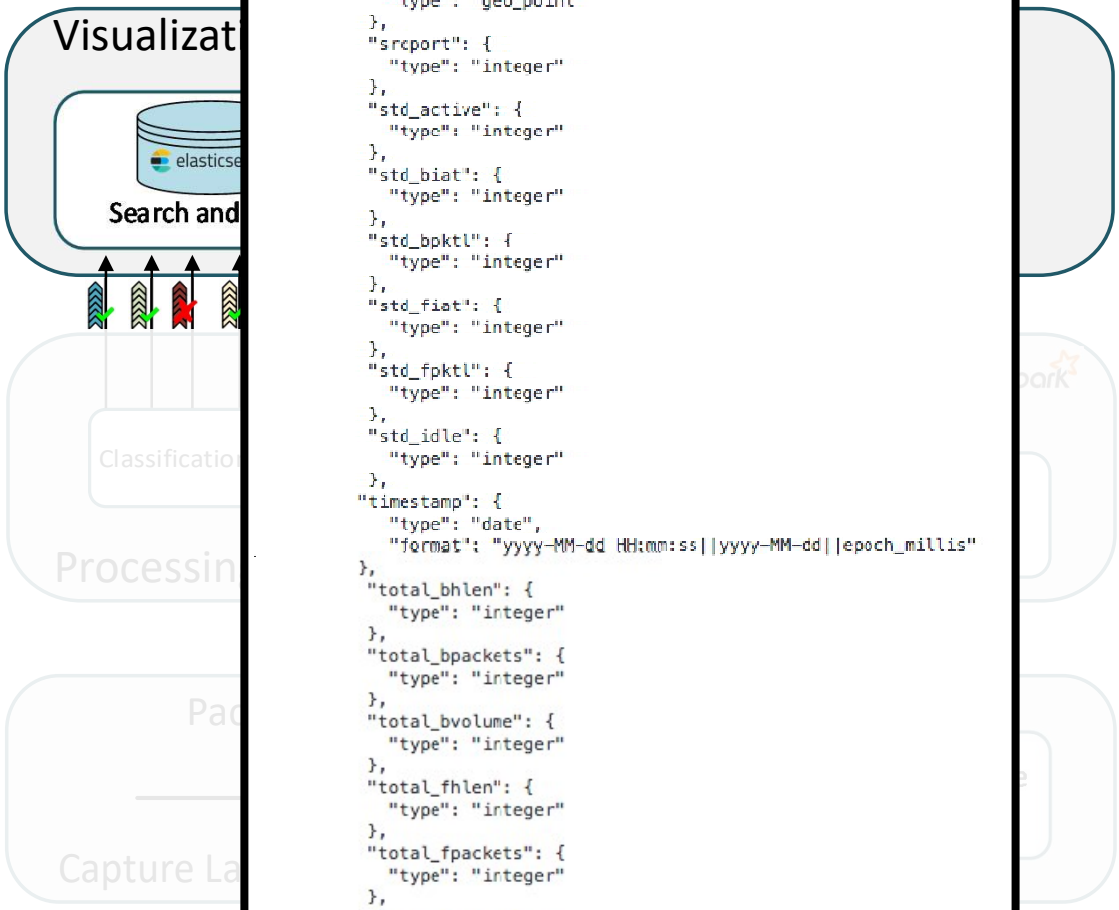



```
PUT spark
{
    "mappings": {
      "test": {
        "properties": {
          "bpsh_cnt": {
            "type": "integer"
          },
          "burg_cnt": {
            "type": "integer"
          },
          "dstip": {
            "type": "ip"
          },
          "dstlocation": {
            "type": "geo_point"
          },
          "dstport": {
            "type": "integer"
          },
          "duration": {
            "type": "float"
          },
          "fpsh_cnt": {
            "type": "integer"
          },
          "furg_cnt": {
            "type": "integer"
          },
          "max_active": {
            "type": "integer"
          },
          "max_biat": {
            "type": "integer"
          },
          "max_bpktl": {
            "type": "integer"
          },
          "max_fiat": {
            "type": "integer"
          },
          "max_fpktl": {
            "type": "integer"
          },
          "max_idle": {
            "type": "integer"
          },
          "mean_active": {
            "type": "integer"
          },
          "mean_biat": {
            "type": "integer"
          },
          "mean_bpktl": {
            "type": "integer"
          },
          "mean_fiat": {
            "type": "integer"
          },
          "mean_fpktl": {
            "type": "integer"
          },
          "mean_idle": {
            "type": "integer"
          },
```

COPPE UFRJ

Arquitetura CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

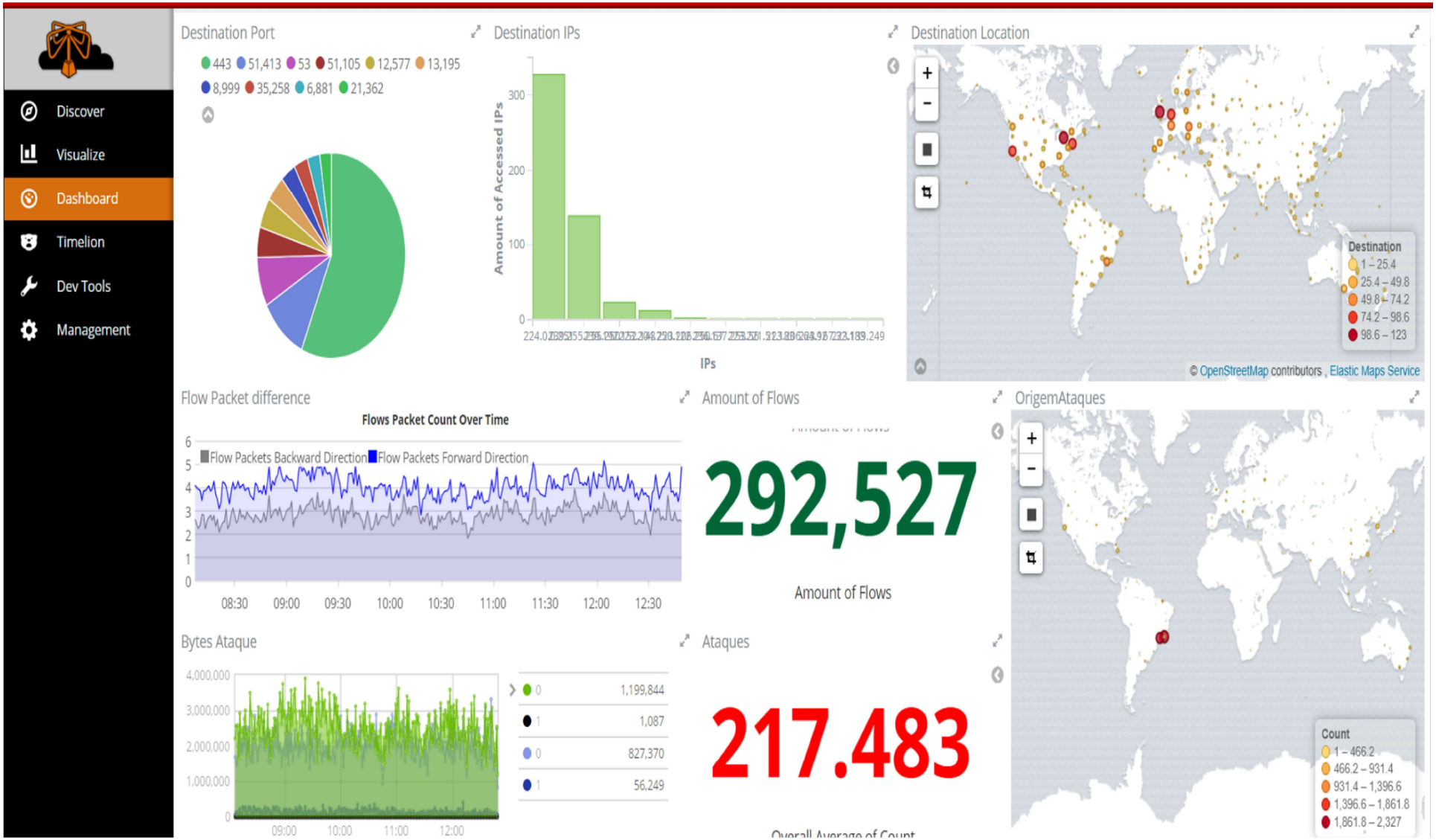



```
      },
      "sflow_bbytes": {
        "type": "integer"
      },
      "sflow_bpackets": {
        "type": "integer"
      },
      "sflow_fbytes": {
        "type": "integer"
      },
      "sflow_fpackets": {
        "type": "integer"
      },
      "srcip": {
        "type": "ip"
      },
      "srclocation": {
        "type": "geo_point"
      },
      "srcport": {
        "type": "integer"
      },
      "std_active": {
        "type": "integer"
      },
      "std_biat": {
        "type": "integer"
      },
      "std_bpktl": {
        "type": "integer"
      },
      "std_fiat": {
        "type": "integer"
      },
      "std_fpktl": {
        "type": "integer"
      },
      "std_idle": {
        "type": "integer"
      },
      "timestamp": {
        "type": "date",
        "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
      },
      "total_bhlen": {
        "type": "integer"
      },
      "total_bpackets": {
        "type": "integer"
      },
      "total_bvolume": {
        "type": "integer"
      },
      "total_fhlen": {
        "type": "integer"
      },
      "total_fpackets": {
        "type": "integer"
      },
      "total_fvolume": {
        "type": "integer"
      }
    }
  }
}
```

COPPE UFRJ

Interface de Visualização CATRACA

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO



DEMONSTRAÇÃO DA FERRAMENTA CATRACA

Offline Classification

Dataset

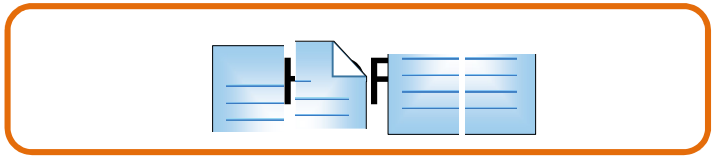
HDFS

Distributed Storage

Offline Classification

Distributed Storage

Offline Classification



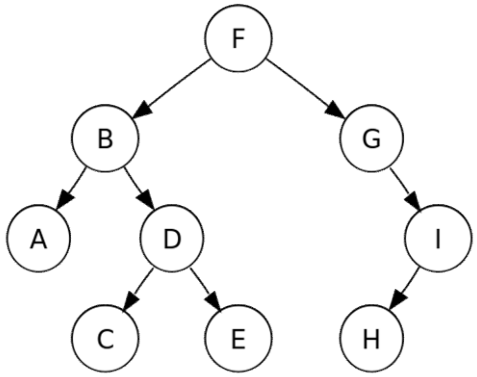
Distributed Storage

Offline Classification

Distributed Processing

Distributed Storage

Offline Classification



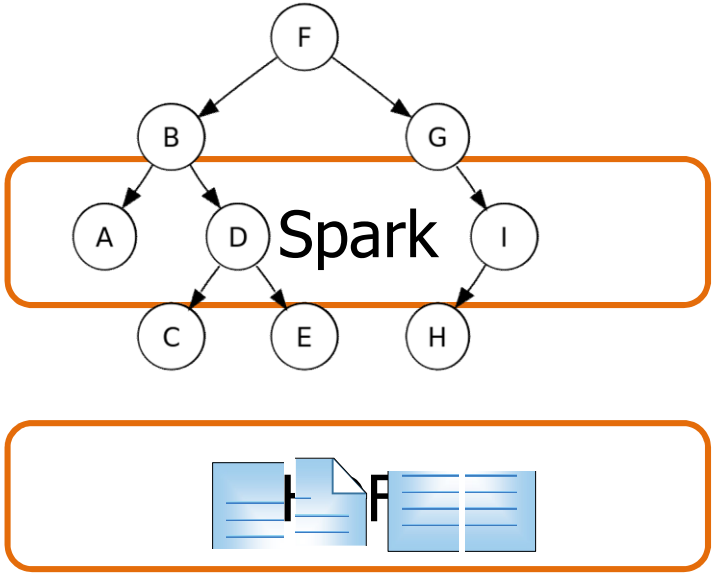
Decision Tree Model

Spark

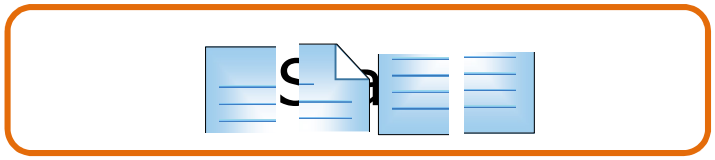
Distributed Processing

Distributed Storage

Offline Classification



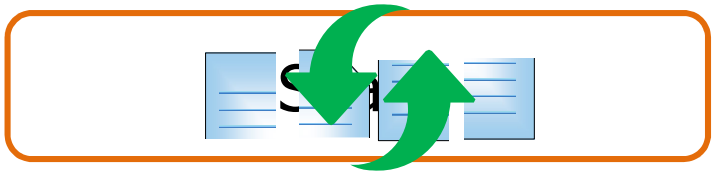
Offline Classification



Distributed Processing

Distributed Storage

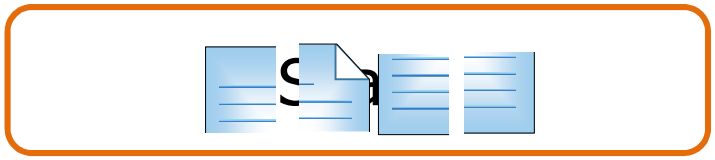
Offline Classification



Distributed Processing

Distributed Storage

Offline Classification



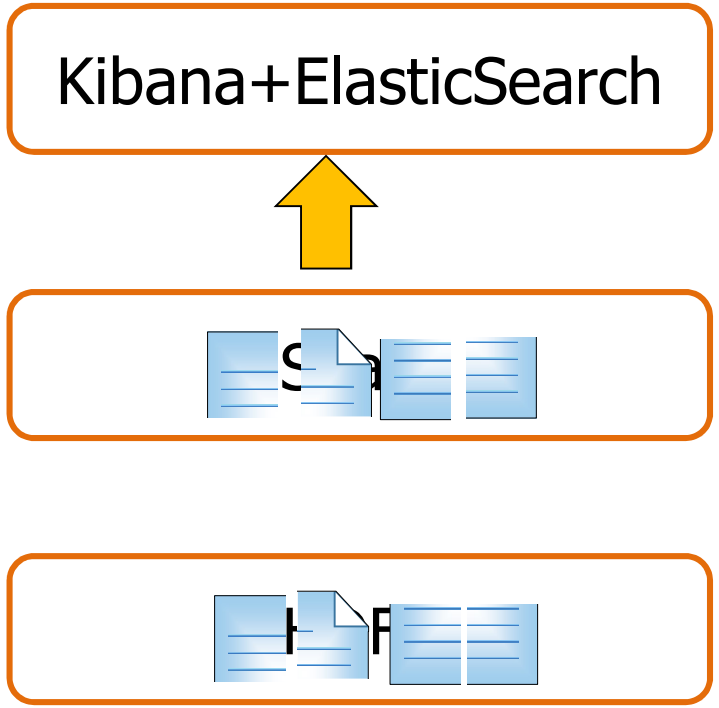
Distributed Processing

Distributed Storage

Offline Classification

	Precision	Sens.	Spec.
Alert	0.933	0.934	0.976
Normal	0.976	0.976	0.934

Results



PARTE V

Considerações Finais e Problemas em Aberto

Conclusão

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Plataformas de processamento distribuído de fluxo
  - Nova direção de pesquisa para detecção de ameaças
  - Grande volume de dados de rede
  - Grande variedade de dados
    - Logs, pacotes, redes sociais
  - Grande velocidade de geração de dados
- Plataformas de Código Aberto
  - Apache Spark
  - Apache Storm
  - Apache Flink

Conclusão

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Arquitetura *lambda* → acurada e rápida
  - Dados em fluxo tratados em tempo quase real
    - Processamento reduzido sobre cada amostr
    - Treinamento e adaptação de modelos são realizados em tempo diferenciado
  - Treinamento prévio
    - Incapacidade de identificar ameaças inéditas.

COPPE UFRJ

Conclusão

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Arquitetura de classificação de tráfego com aprendizado adaptativo
  - Identificar ameaças inéditas (*zero-day threats*)
  - Aperfeiçoamento gradual do modelo conforme novas amostras rotuladas surgem
  - Dados rotulados recebidos em tempo real
    - Potes de mel (*honeypots*) → captura do comportamento das ameaças
  - Introdução de um custo maior de processamento por amostra
    - Passo de adaptação para os algoritmos incrementais

Detecção de Intrusão por Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- CATRACA → Estudo de Caso da Aula Prática
  - Ferramenta de Detecção de Intrusão baseada em Aprendizado de Máquina executado sobre a plataforma de processamento por fluxo Apache Spark
- Apache Metron
  - Arcabouço de análise de segurança baseado no processamento e correlação de grandes massas de dados
    - Fontes diversas de dados → Correlação de eventos complexos
    - Apache Storm
- Apache Spot
  - Projeto em incubação no Apache Foudation
    - Telemetria + Aprendizado de Máquina → Detecção de Ameaças

COPPE UFRJ

Detecção de Intrusão por Aprendizado de Máquina

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Stream4Flow
  - Monitoramento do comportamento da rede através de Apache Spark e ElasticSearch → Visualização
- Hogzilla
  - Sistema de detecção de intrusão com suporte para Snort, Sflow, GrayLog, Apache Spark, Hbsae e libnDPI
    - Processamento de grandes massas de dados e visualização de tráfego da rede

Perspectivas Futuras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Desafio → detecção de anomalias através do processamento distribuído de fluxos de dados é um desafio
  - Dados gerados de forma contínua e com alta velocidade
  - Algoritmos que processam fluxos → soluções aproximadas e resposta rápida usando poucos recursos de memória
- **Pesquisas atuais** em processamento de fluxos de dados
  - Busca por algoritmos e estruturas de dados com margens estreitas de erro
  - fazer análises em dados criptografados ainda é um desafio
  - Questões semânticas→ operadores aplicados sobre os dados

Perspectivas Futuras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- A taxa de geração dos fluxos varia → Valores elevados valores → estouro da fila de processamento dos sistemas de detecção de ameaças
  - **Pesquisas atuais:** Diminuição da representação dos dados ao invés de simplesmente descartar as amostras quando a fila estiver cheia
    - Técnicas como amostragem
    - Descarte de carga (*Load Shedding*)
    - Estruturas de dados de Sinopse (*Synopsis*)

Trabalhos Futuros

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

- Algoritmos de aprendizado incremental
  - Atualização das estatísticas para classificação com a chegada de cada nova amostra
  - Comportamento estacionário
    - Mudança → classificação com baixa acurácia
      - *Concept drift*
      - Desenvolvimento de algoritmos de aprendizado de máquina resistentes ao *concept drift*
- Aprendizado incremental por agregados de classificadores
  - Classificação através de diversos classificadores de baixa acurácia e alta variabilidade → Classificação final através de um esquema de votação
  - Aglomerado de classificadores → resistente a mudanças no comportamento das amostras

GTA / UFRJ
GRUPO DE TELEINFORMÁTICA E AUTOMAÇÃO

Aprendizado de Máquina em Plataformas de Processamento Distribuído de Fluxo: Análise e Detecção de Ameaças em Tempo Real

Martin Andreoni Lopez (UFRJ), Igor Jochem Sanz (UFRJ), Antonio G. P. Lobato (UFRJ), Diogo M. F. Mattos (UFF) e Otto Carlos M. B. Duarte (UFRJ)

XXXVI Simpósio Brasileiro de Redes de Computadores

09/05/2018 – Campos do Jordão

Programa de Engenharia Elétrica - PEE/COPPE/UFRJ
Universidade Federal do Rio de Janeiro

COPPE 50 UFRJ