Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

Benchmarking de Plataformas Blockchain

Ana Catarina Dinis Diegues

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas de Informação orientada pelo Professor Doutor Paulo Rupino da Cunha da Faculdade de Ciências e Tecnologia, e pelo Professor Doutor Manuel Paulo de Albuquerque Melo da Faculdade de Economia e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Setembro de 2020

UNIVERSIDADE D
COIMBRA

Esta página foi propositadamente deixada em branco.

Abstract

When it comes to understanding which blockchain platforms is best suited to existing needs, it is necessary to evaluate its performance in different environments that may arise. This assessment can be achieved using benchmark testing tools that can study these platforms.

Throughout this document, a study is carried out on the theory underlying blockchain technology and the various platforms that implement it. Existing studies in the area of benchmarking of blockchain platforms were also collected and analyzed, which were fundamental both for the choice of the benchmark tool to be used and for understanding how the benchmark itself was carried out.

In order to assess the performance of the blockchain platforms, in this work, a system was implemented that could analyze the Hyperledger Fabric and FISCO BCOS platforms. This system is based on the Hyperledger Caliper tool. From the tests performed, it was possible to collect metrics related to latency, throughput, send rate, execution time, CPU and RAM consumption and the number of successful transactions.

Analyzing the results obtained, it is possible to conclude, in general, that the behaviors of the studied blockchain platforms are very different. Hyperledger Fabric has a more constant and stable performance than FISCO BCOS. It is verified that the type of operation that is performed, the use of a consensus mechanism and the use of different databases are factors that directly influence the performance of the Fabric platform.

Keywords

Blockchain Platforms, *Distributed Ledger Technology*, *Smart Contracts*, *Consensus Mechanism*, *Benchmark*, *Hyperledger Fabric*, FISCO BCOS, *Hyperledger Caliper*.

Esta página foi propositadamente deixada em branco.

Resumo

Quando se trata de perceber qual a plataforma *blockchain* que melhor se adequa às necessidades existentes, é necessário avaliar o seu desempenho em diversos ambientes que possam surgir. Esta avaliação pode ser conseguida recorrendo a ferramentas de testes de *benchmark* que consigam estudar estas plataformas.

Ao longo deste documento é realizado um estudo acerca da teoria subjacente à tecnologia *blockchain* e das várias plataformas que a implementam. Foram também recolhidos e analisados estudos existentes na área do *benchmarking* de plataformas *blockchain* que foram fundamentais tanto para a escolha da ferramenta de *benchmark* a utilizar como para perceber como era realizado o *benchmark* em si.

Com o propósito de proceder à avaliação do desempenho das plataformas *blockchain*, neste trabalho, foi implementado um sistema que conseguisse analisar as plataformas *Hyperledger Fabric* e FISCO BCOS. Este sistema tem como base a ferramenta *Hyperledger Caliper*. Dos testes executados foi possível recolher métricas relativas à latência, *throughput*, *send rate*, tempo de execução, consumo de CPU e de memória RAM e o número de transações bem sucedidas.

Analisando os resultados obtidos, é possível concluir, de uma forma geral, que os comportamentos das plataformas *blockchain* estudadas são muito distintos. Sendo que a *Hyperledger Fabric* apresenta um desempenho mais constante e estável que o da FISCO BCOS. É verificado que o tipo de operação que é realizada, o uso de mecanismo de consenso e a utilização de diferentes bases de dados são factores que influenciam diretamente a performance da plataforma *Fabric*.

Palavras-Chave

Plataformas *Blockchain*, Tecnologia de *Ledgers* Distribuídos, *Smart Contracts*, Mecanismo de Consenso, *Benchmark*, *Hyperledger Fabric*, FISCO BCOS, *Hyperledger Caliper*.

Esta página foi propositadamente deixada em branco.

Agradecimentos

A realização deste trabalho não seria possível sem o apoio e acompanhamento fornecidos por várias pessoas ao longo de todo o percurso.

Aos meus orientadores de dissertação, Professor Doutor Paulo Rupino da Cunha, do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, e Professor Doutor Manuel Paulo de Albuquerque Melo, da Faculdade de Economia da Universidade de Coimbra, o meu mais sincero obrigado por toda a ajuda, apoio e disponibilidade que sempre demonstraram.

Aos meus pais e ao meu irmão, por todo o carinho que sempre me dedicaram e por toda a paciência e força que me deram para que me mantivesse sempre motivada a fazer mais e melhor.

Aos meus amigos por sempre me escutarem quando as coisas corriam pior e incentivarem a continuar.

A todos, um muito obrigado!

Esta página foi propositadamente deixada em branco.

Conteúdo

Esta página foi propositadamente deixada em branco.

Acrónimos

BFT Byzantine Fault Tolerance. 27

CFT Crash Fault Tolerance. 27

CPU Central Process Unit. 40, 42, 51, 53–55, 66, 68, 79

DEI Departamento de Engenharia Informática. 47

DLT Distributed Ledger Technology. 39

IBFT Istanbul Byzantine Fault Tolerance. 25, 27, 39

IPN Instituto Pedro Nunes. 2

PoW Proof-of-Work. 25

RAM Random Access Memory. 2, 42, 47

RPCs Remote Procedure Calls. 24

SUT System Under Test. 37, 38

TPS Transações Por Segundo. 48, 50–55, 57, 60, 62, 64, 66, 68, 71–73, 75, 77, 81–83, 85–87, 92, 93, 95, 97, 99, 101, 103, 107, 109, 110, 113, 115, 116

Esta página foi propositadamente deixada em branco.

Lista de Figuras

Esta página foi propositadamente deixada em branco.

Lista de Tabelas

Esta página foi propositadamente deixada em branco.

Capítulo 1

Introdução

O presente documento surge do trabalho realizado no âmbito da Dissertação "*Benchmarking* de Plataformas *Blockchain*", inserida no Mestrado de Engenharia Informática, do Departamento de Engenharia Informática, da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, no ramo de Sistemas de Informação. Este teve início em Setembro de 2019 e é relativo ao ano letivo 2019/2020.

Esta Dissertação é orientada pelo Professor Doutor Paulo Rupino da Cunha, da Faculdade de Ciências e Tecnologia da Universidade de Coimbra, e pelo Professor Doutor Manuel Paulo de Albuquerque Melo, da Faculdade de Economia da Universidade de Coimbra. Esta é apresentada ao Departamento de Engenharia Informática, da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Este capítulo encontra-se divido em quatro secções: a secção 1.1 - que se refere a uma breve introdução da tecnologia *blockchain* e onde é apresentada a motivação que levou à realização desta dissertação; a secção 1.2 - em que se descreve o trabalho previamente realizado no ano letivo 2018/2019; a secção 1.3 - onde são apresentados os principais objetivos deste trabalho; e, por fim, a secção 1.4 - que diz respeito à estrutura definida para este documento.

1.1 Contexto e Motivação

Uma *blockchain* corresponde a um *ledger* digital resistente a violações, implementado de forma distribuída - sem um repositório central - e, geralmente, sem uma autoridade central. Ela permite que os utilizadores de uma comunidade registem transações num livro partilhado dentro desta, sendo que nenhuma transação pode ser alterada depois de publicada (Yaga et al., 2018). Uma *blockchain* é um tipo especial de base de dados, na qual estão inseridos blocos estruturados sequencialmente de forma cronológica, conectados entre si. Cada um destes contém as várias transações que foram validadas e que serviram para construir o respetivo bloco. Assim, é possível obter um registo de dados imutável, dada a conexão existente entre estes, que se encontra distribuída por vários nós presentes na rede (Antunes, 2019).

Ao longo do tempo, surgiram várias plataformas que têm como base a tecnologia *blockchain*. Cada uma destas apresenta diferentes caraterísticas, tais como o suporte para aplicações descentralizadas e a utilização de *smart contracts* - que consiste num programa inserido na *blockchain* que define as regras para o envio de transações (Gupta, 2017). A

indústria começou a revelar um interesse especial por estas caraterísticas únicas, por ser notório que a utilização destas plataformas ia muito além da criptomoeda. Diversos autores sugeriram então diversas aplicações desta tecnologia em diversos setores da indústria, como é o caso do controlo do comércio de diamantes de sangue (Minkenberg, 2015), na melhoria dos processos da logística marítima (Gronholt-Pedersen, 2018) e para revolucionar a indústria musical (Heap, 2017).

Com a finalidade de avaliar e comparar o desempenho de diferentes plataformas *blockchain* foi realizado este trabalho, que proporcionou um aumento do conhecimento de como esta tecnologia funciona.

Com esta "Dissertação/Estágio em Sistemas de Informação"pretende-se dar continuidade ao trabalho apresentado no ano letivo 2018/2019, (Dias, 2019).

1.2 Trabalho Previamente Realizado

O trabalho realizado no ano letivo 2018/2019, (Dias, 2019), foi realizado no âmbito de estudar e comparar diferentes plataformas *blockchain*.

Como alvo de estudo do seu trabalho, (Dias, 2019) escolheu as plataformas *Hyperledger Fabric*, na versão 1.4, e a *Quorum*, na versão 2.0. Contudo, o estudo desta última não foi possível, pois a conexão desta com a ferramenta de testes de *benchmark* não foi conseguida.

A ferramenta de testes de *benchmark* escolhida por (Dias, 2019), para proceder à avaliação da plataforma *blockchain*, foi a *Gauge*. Esta ferramenta consistia num *fork* da *Hyperledger Caliper*, com a diferença de, alegadamente, suportar a plataforma *Quorum*. Em 2019, a *Gauge* deixou de ser acessível, ficando descontinuada, não sendo possível a sua utilização.

Para a elaboração dos testes, (Dias, 2019) utilizou 6 máquinas virtuais presentes no Instituto Pedro Nunes (IPN), cada uma delas com 80 GB de memória em disco, 8 GB de Random Access Memory (RAM) e 4 vCPU's, correndo o sistema operativo *Debian* 9.

A ferramenta *Gauge* estava preparada com testes exemplo bastante específicos relativos à *Hyperledger Fabric*. Desta forma, foi possível ao autor realizar um estudo mais aprofundado desta plataforma. A estrutura e funcionamento desta ferramenta são apresentadas posteriormente na secção 4.2.3.

Para a execução dos testes, foi utilizada uma rede *blockchain* da qual fazia parte um *Solo Orderer*, duas organizações (Org1 e Org2), com dois *peers* cada uma, a comunicarem através de um canal único, utilizando *LevelDB* para a persistência de dados no *ledger*.

A figura 1.1 representa o sistema de *benchmarking* utilizado por (Dias, 2019).

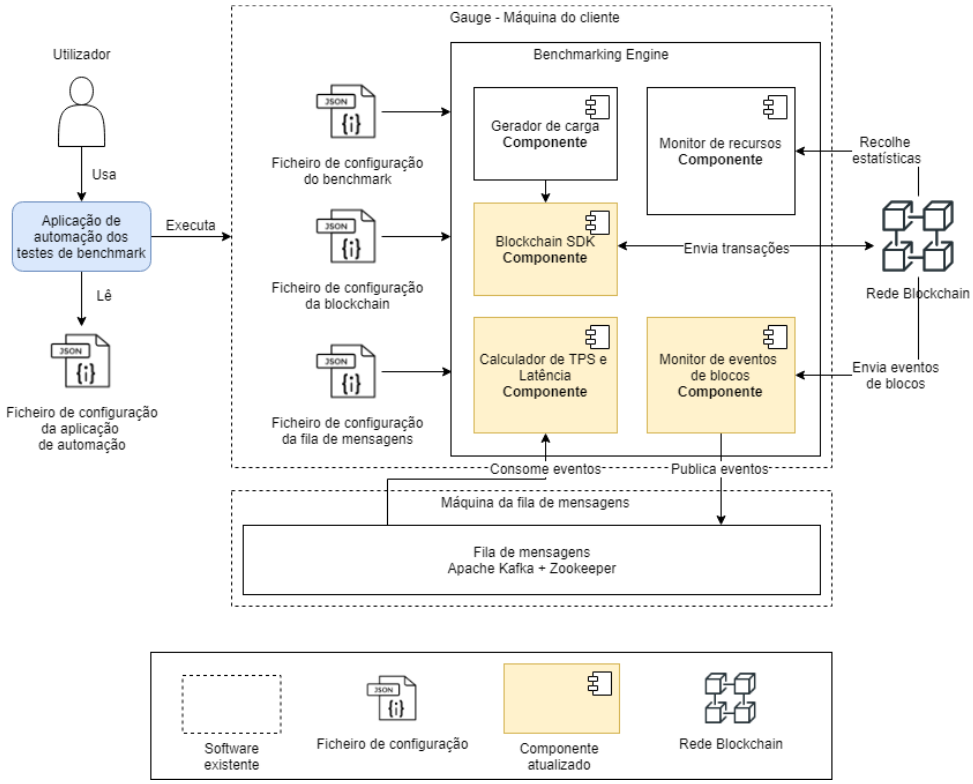


Figura 1.1: Sistema de *benchmarking*, retirado de (Dias, 2019)

De seguida são apresentadas as conclusões gerais da execução dos testes de *benchmark* alcançadas por (Dias, 2019):

- A arquitetura e o modelo de transações da plataforma *Hyperledger Fabric* proporciona um melhor desempenho para operações de leitura sobre a *blockchain* quando comparadas com as operações de escrita. As operações de leitura são praticamente imediatas, enquanto que as operações de escrita passam pelas fases *execute-order-validate*, reduzindo o *throughput* e aumentando a latência da plataforma (este funcionamento encontra-se explicado na secção 3.5.1);
- Nos testes desenhados para medir os valores máximos de *throughput* e latência da plataforma, quando esta é submetida a operações de escrita sobre a *blockchain*, verificou-se que eram atingidos os seus valores máximos no intervalo entre as 3500 e as 4000 transações;
- A modificação dos parâmetros do *Orderer* - componente responsável pelo ordenamento das transações recebidas-, tal como o tamanho do bloco gerado, demonstrou aumentar a performance da plataforma estudada;
- Relativamente à escalabilidade da plataforma, não foi possível recolher resultados fiáveis.

O autor termina o seu documento sugerindo outras abordagens para a realização de trabalhos semelhantes, tais como:

- Numa primeira abordagem ao trabalho desenvolvido, o autor (Dias, 2019) propõe redesenhar os *smart contracts* que implementam os testes de operações de escrita na

blockchain, fazendo-lhes um *update* para funções que requeiram um trabalho computacional maior;

- Outra sugestão deixada pelo autor passa por executar os testes realizados por este utilizando uma combinação da rede *Hyperledger Fabric* diferente;
- Propõe ainda a substituição de *LevelDB* por *CouchDB* para a persistência de dados, de forma a verificar como a base de dados utilizada pode influenciar a performance da plataforma;
- Por fim, deixa a sugestão de terminar o *update* do componente *Blockchain SDK* para a comunicação com a plataforma *Quorum*, iniciado por ele, na ferramenta *Gauge*.

Menciona, por fim, algumas das dificuldades que enfrentou, como por exemplo, o facto de ser uma tecnologia recente e, por essa razão, existir pouca informação para resolver possíveis problemas que apareceram.

1.3 Objetivos

Tendo em consideração que se trata de um trabalho de continuidade referente ao ano letivo 2018/2019, foi tido em conta todo o desenvolvimento deste.

O principal objetivo deste trabalho é comparar diferentes plataformas *blockchain*, utilizando para isso uma ferramenta de testes de *benchmark*.

Para alcançar este objetivo foi realizado, em primeiro lugar, um estudo acerca dos conceitos base da tecnologia *Blockchain* e das diversas definições a ela associadas. Desta forma, foi possível obter uma visão geral acerca do seu funcionamento. De seguida, procedeu-se à investigação de plataformas *blockchain* e de trabalhos realizados no âmbito do *benchmarking* destas. Posteriormente, foi construído o ambiente de testes, necessário para a elaboração deste trabalho, utilizando a ferramenta *Hyperledger Caliper*.

O objetivo inicial desta dissertação consistia na avaliação do desempenho de duas plataformas *blockchain* privadas, a *Quorum* (JPMorgan Chase & Co., 2016) e a *Hyperledger Fabric* (The Linux Foundation, 2018), refazendo os testes implementados no trabalho anterior de (Dias, 2019) referentes a esta última.

Relativamente à plataforma *Quorum*, não foi possível a sua avaliação, visto que esta não é suportada pela *Hyperledger Caliper* (The Linux Foundation, 2019). Assim sendo, foi decidido estudar a plataforma FISCO BCOS (WeBank et al., 2017) e a *Hyperledger Fabric*, ambas suportadas pela ferramenta de *benchmark* escolhida.

1.4 Estrutura do Documento

Tendo sido apresentados os principais objetivos deste trabalho, bem como a motivação para o seu desenvolvimento, o resto do relatório foi dividido da seguinte forma:

O **capítulo 2** representa o planeamento do trabalho realizado no primeiro e segundo semestres, bem como as respetivas justificações para os desvios que ocorreram.

No **capítulo 3** são descritos alguns conceitos fundamentais subjacentes à tecnologia *Blockchain*, utilizados ao longo da dissertação.

O **capítulo 4** diz respeito à recolha de trabalhos elaborados no âmbito do *benchmarking* de plataformas *blockchain* e à comparação e escolha da ferramenta de testes de *benchmark* a utilizar.

No **capítulo 5** é descrito o ambiente de testes implementado, as suas principais caraterísticas e a sua organização. É feita também a definição dos testes.

O **capítulo 6** é composto pelos resultados provenientes dos testes realizados bem como a sua análise e respetivas conclusões.

Por fim, o **capítulo 7** refere-se às conclusões gerais de todo o trabalho elaborado e possíveis sugestões para trabalho futuro.

Esta página foi propositadamente deixada em branco.

Capítulo 2

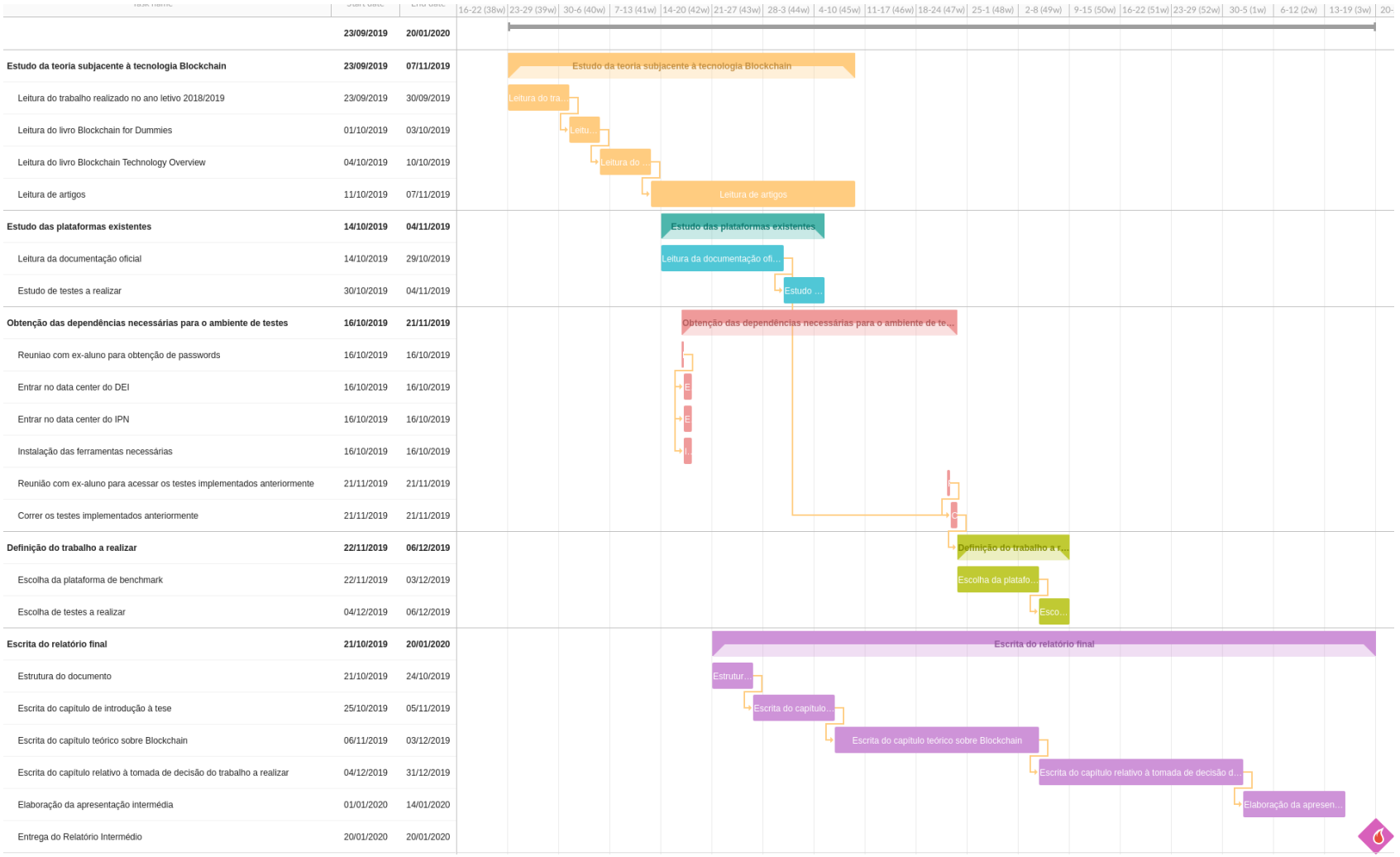
Planeamento

Neste capítulo é apresentado e descrito o planeamento do trabalho. Para uma melhor compreensão, este foi dividido em duas partes, referentes ao primeiro e segundo semestre. Como nem sempre o trabalho planeado corre como esperado, podendo demorar mais ou menos tempo que o previsto, mostram-se dois Diagramas de Gantt para cada um dos semestres, em que o primeiro corresponde ao planeamento e o segundo diz respeito ao que efetivamente aconteceu.

De forma a perceber em que consiste cada tarefa inserida no Diagrama de Gantt, bem como os problemas e/ou desvios encontrados, é explicada cada tarefa individualmente.

2.1 Planeamento do Primeiro Semestre

Na presente secção procede-se à apresentação do trabalho planeado e realizado no primeiro semestre. São descritas detalhadamente cada uma das tarefas principais e justificados os respetivos desvios temporais presentes em cada um dos Diagramas de Gantt. Nas figuras 2.1 e 2.2 encontra-se representado o trabalho planeado e executado, respetivamente, para o primeiro semestre. As principais tarefas que constituem os Diagramas, bem como a explicações dos desvios ocorridos, são explicadas de seguida, em forma de tabela.



	23/09/2019	**20/01/2020**
Estudo da teoria subjacente à tecnologia Blockchain	**23/09/2019**	**07/11/2019**
Leitura do trabalho realizado no ano letivo 2018/2019	23/09/2019	30/09/2019
Leitura do livro Blockchain for Dummies	01/10/2019	03/10/2019
Leitura do livro Blockchain Technology Overview	04/10/2019	10/10/2019
Leitura de artigos	11/10/2019	07/11/2019
Estudo das plataformas existentes	**14/10/2019**	**04/11/2019**
Leitura da documentação oficial	14/10/2019	29/10/2019
Estudo de testes a realizar	30/10/2019	04/11/2019
Obtenção das dependências necessárias para o ambiente de testes	**16/10/2019**	**21/11/2019**
Reuniao com ex-aluno para obtenção de passwords	16/10/2019	16/10/2019
Entrar no data center do DEI	16/10/2019	16/10/2019
Entrar no data center do IPN	16/10/2019	16/10/2019
Instalação das ferramentas necessárias	16/10/2019	16/10/2019
Reunião com ex-aluno para acessar os testes implementados anteriormente	21/11/2019	21/11/2019
Correr os testes implementados anteriormente	21/11/2019	21/11/2019
Definição do trabalho a realizar	**22/11/2019**	**06/12/2019**
Escolha da plataforma de benchmark	22/11/2019	03/12/2019
Escolha de testes a realizar	04/12/2019	06/12/2019
Escrita do relatório final	**21/10/2019**	**20/01/2020**
Estrutura do documento	21/10/2019	24/10/2019
Escrita do capítulo de introdução à tese	25/10/2019	05/11/2019
Escrita do capítulo teórico sobre Blockchain	06/11/2019	03/12/2019
Escrita do capítulo relativo à tomada de decisão do trabalho a realizar	04/12/2019	31/12/2019
Elaboração da apresentação intermédia	01/01/2020	14/01/2020
Entrega do Relatório Intermédio	20/01/2020	20/01/2020

Figura 2.1: Diagrama de Gantt do trabalho planeado para o primeiro semestre

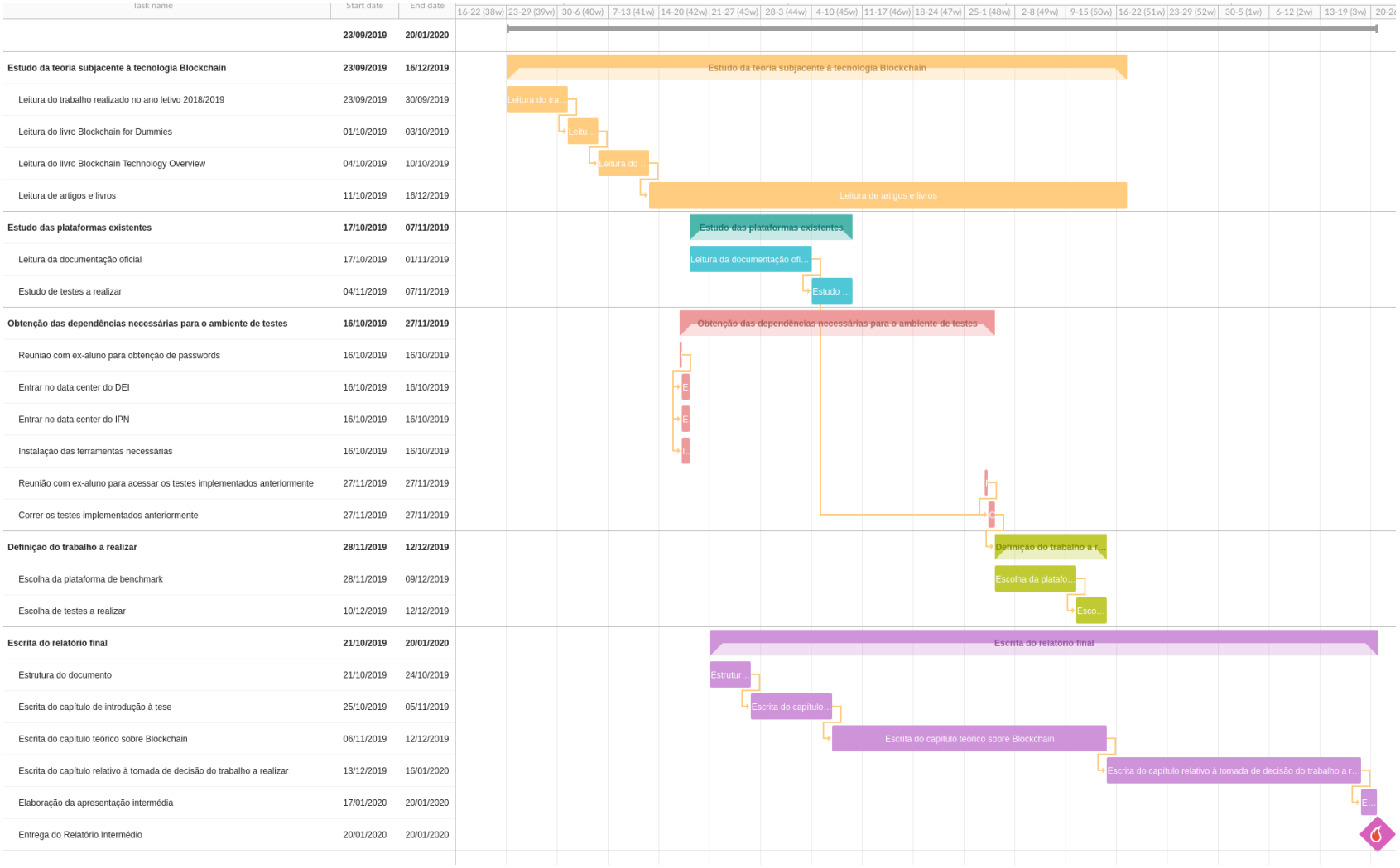


Figura 2.2: Diagrama de Gantt do trabalho realizado no primeiro semestre

Tarefa 1 - Estudo da teoria subjacente à tecnologia *Blockchain*

A primeira tarefa representa o estudo dos conceitos fundamentais para compreender o funcionamento da tecnologia *Blockchain*. A recolha desta informação foi conseguida através da leitura de *papers* e livros de diversos autores. Como é possível observar na tabela 2.1, a data final planeada para completar esta tarefa é distinta da executada, pois, à medida que a escrita do relatório intermédio evoluiu, tornou-se necessário reler alguns conceitos e pesquisar acerca de outros que foram surgindo. Esta tarefa demorou mais um mês e nove dias do que estava previsto.

Tempo	Data de inicio	Data de fim
Planeado	23/09/2019	07/11/2019
Executado	23/09/2019	16/12/2019

Tabela 2.1: Estudo da teoria subjacente à tecnologia *Blockchain*

Tarefa 2 - Estudo das plataformas existentes

A segunda tarefa diz respeito ao estudo das plataformas *blockchain* através da leitura da sua documentação oficial e de possíveis testes a realizar sobre estas. Observando a tabela 2.2, é possível notar um desvio de três dias na data de início desta tarefa, que se deveu ao atraso na leitura da teoria subjacente à tecnologia *blockchain*. Esta tarefa foi fundamental para compreender o funcionamento das plataformas que se pretendia estudar, a *Hyperledger Fabric* e a *Quorum*, de forma a ficar a conhecer a sua arquitetura e as suas propriedades.

Tempo	Data de inicio	Data de fim
Planeado	14/10/2019	04/11/2019
Executado	17/10/2019	07/11/2019

Tabela 2.2: Estudo das plataformas existentes

Tarefa 3 - Obtenção das dependências necessárias para o ambiente de testes

A terceira tarefa presente no plano de trabalho consiste na obtenção das dependências necessárias para o ambiente de testes. Para tal, procedeu-se a uma reunião com o ex-aluno Rui Dias no dia 16/10/2019, na qual foram instaladas todas as ferramentas necessárias para executar os testes por ele implementados bem como obtidas as *passwords* por ele definidas. Após completar a leitura da documentação oficial, foi possível marcar uma nova reunião, realizada no dia 27/11/2019 (ao início esta seria no dia 21/11/2019, o que explica a diferença de sete dias entre as datas de fim apresentadas na tabela 2.3). Nesta foi conseguida a execução dos testes implementados no ano letivo anterior. Esta tarefa apresenta a duração de um mês por incluir várias sub-tarefas que se encontram espaçadas entre si. O trabalho efectuado nesta tarefa, na realidade, apenas durou cerca de 2 dias, que consistiram nos dias das reuniões com o ex-aluno.

Tempo	Data de inicio	Data de fim
Planeado	16/10/2019	21/11/2019
Executado	16/10/2019	27/11/2019

Tabela 2.3: Obtenção das dependências necessárias para o ambiente de testes

Tarefa 4 - Definição do trabalho a realizar

A tabela 2.4 refere-se à quarta tarefa. Esta passa pela tomada de decisão sobre o trabalho a realizar no segundo semestre. Consistiu na escolha da plataforma *blockchain* a estudar, bem como a ferramenta de *benchmark* a utilizar. Nesta fase foi decidido estudar as plataformas *Hyperledger Fabric* e *Quorum*, utilizando a ferramenta *Gauge*.

Tempo	Data de inicio	Data de fim
Planeado	22/11/2019	06/12/2019
Executado	28/11/2019	12/12/2019

Tabela 2.4: Definição do trabalho a realizar

Tarefa 5 - Escrita do relatório intermédio

Por fim, a quinta tarefa consistiu na elaboração e escrita do relatório intermédio. Esta passou pela definição da sua estrutura e pela escrita dos capítulos teóricos relativos à tecnologia *blockchain*. Apesar de não ser notório na tabela 2.5, existiram atrasos significativos na escrita do relatório intermédio devido à dificuldade de conciliar esta com os trabalhos e exames a realizar nas outras unidades curriculares frequentadas.

Tempo	Data de inicio	Data de fim
Planeado	21/10/2019	20/01/2020
Executado	21/10/2019	20/01/2020

Tabela 2.5: Escrita do relatório intermédio

2.2 Planeamento do Segundo Semestre

Nesta secção é apresentado o planeamento do trabalho a realizar no segundo semestre bem como aquele que efetivamente foi realizado. Os Diagramas de Gantt representados nas figuras 2.3 e 2.4 dizem respeito ao trabalho planeado e executado, respetivamente. É possível observar a adição de diversas tarefas visto que houve a necessidade de as efectuar aquando da realização deste trabalho e de repetir outras que já haviam sido feitas no primeiro semestre. Esta repetição foi necessária por se verificar que existia outra ferramenta de testes de *benchmark* que se adequava mais para a elaboração deste trabalho e, consequentemente, ao alterar a ferramenta, foi necessário avaliar quais as plataformas *blockchain* que esta permite estudar.

As tarefas principais que integram os Diagramas, bem como a sua definição e explicação dos problemas que ocorreram, são explicadas de seguida, em forma de tabela.

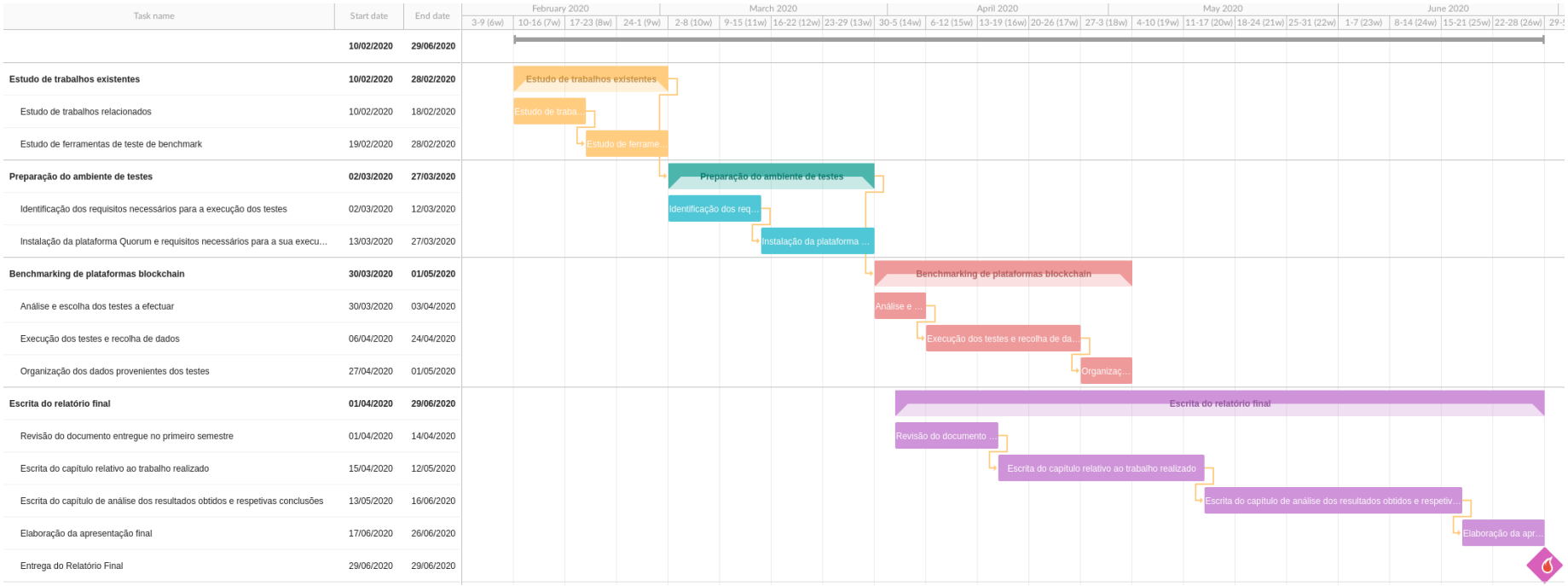


Figura 2.3: Diagrama de Gantt do trabalho planeado para o segundo semestre

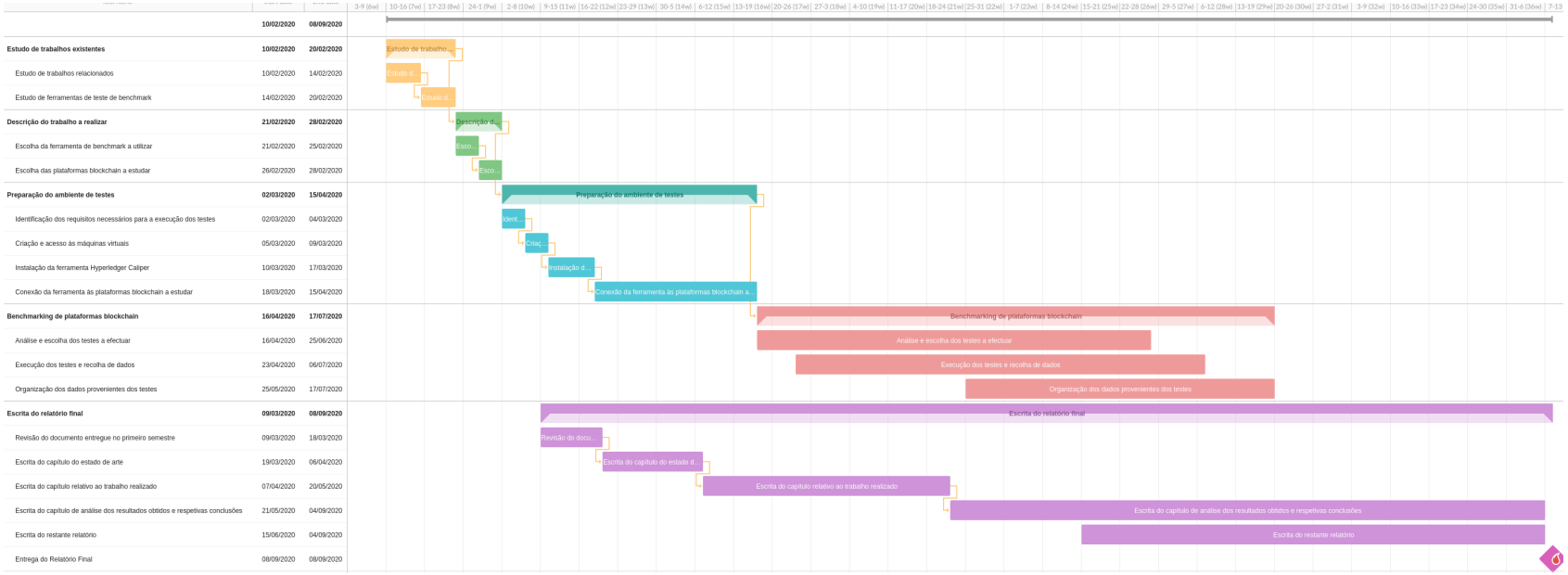


Figura 2.4: Diagrama de Gantt do trabalho realizado no segundo semestre

Tarefa 1 - Estudo de trabalhos existentes

A primeira tarefa representa a recolha de informação através de trabalhos relacionados, de forma a encontrar diversos trabalhos de *benchmark* de plataformas *blockchain*. Estes dados são importantes para verificar se os resultados obtidos vão de encontro às conclusões retiradas pelos autores e para avaliar quais as métricas que é costume recolher. Nesta tarefa foram também pesquisadas e comparadas ferramentas de *benchmark* que pudessem ser utilizadas para a realização do trabalho. Como demonstra a tabela 2.6, esta tarefa demorou menos tempo que o planeado (oito dias), visto que no primeiro semestre já tinha sido feita a recolha destes trabalhos então, neste segundo semestre, apenas foi necessária a sua leitura e análise.

Tempo	Data de inicio	Data de fim
Planeado	10/02/2020	28/02/2020
Executado	10/02/2020	20/02/2020

Tabela 2.6: Estudo de trabalhos existentes

Tarefa 2 - Descrição do trabalho a realizar

A segunda tarefa refere-se à tomada de decisão sobre o trabalho a realizar. Esta tarefa não fazia parte do diagrama inicial, contudo, após fazer uma análise e comparação das ferramentas existentes, foi necessário reavaliar a escolha feita no primeiro semestre. Inicialmente, tinha sido escolhida a ferramenta de *benchmarking Gauge*, porém optou-se por mudar para a *Hyperledger Caliper*, por se adequar melhor às necessidades deste trabalho. Esta alteração levou ao repensar das plataformas *blockchain* a estudar, visto que cada uma das ferramentas suporta plataformas *blockchain* distintas. Nesta fase foi decidido estudar as plataformas *Hyperledger Fabric* e FISCO BCOS. Na tabela 2.7 encontra-se a duração desta tarefa.

Tempo	Data de inicio	Data de fim
Planeado	——	——
Executado	21/02/2020	28/02/2020

Tabela 2.7: Planeamento do trabalho a realizar

Tarefa 3 - Preparação do ambiente de testes

A terceira tarefa consiste na identificação dos requisitos necessários para executar a ferramenta de *benchmark* bem como para fazer a sua conexão com as plataformas *blockchain* a estudar. Após criadas as máquinas virtuais, procedeu-se então à instalação da *Hyperledger Caliper* em cada uma delas e à ligação com a plataforma pretendida. Observando a tabela 2.8 é possível verificar que esta tarefa demorou muito mais tempo que o esperado (18 dias de diferença), pois existiram diversas dificuldades, tanto na instalação da ferramenta como na conexão com as plataformas. Foi no decorrer desta tarefa que se experimentou a conexão da ferramenta com um adaptador da plataforma *Quorum*, contudo não foi possível concluir esta tarefa devido ao aparecimento de vários problemas para os quais não se encontrou solução. Após esta tentativa, foi decidido trocar o estudo desta plataforma pela análise da plataforma FISCO BCOS.

Tempo	Data de inicio	Data de fim
Planeado	02/03/2020	27/03/2020
Executado	02/03/2020	15/04/2020

Tabela 2.8: Preparação do ambiente de testes

Tarefa 4 - *Benchmarking* de plataformas *blockchain*

A quarta tarefa diz respeito à análise dos testes que a ferramenta *Hyperledger Caliper* permite realizar e à escolha de quais executar. Nesta tarefa verificou-se que a ferramenta se encontra um pouco limitada no que diz respeito à avaliação de plataformas *blockchain*, apesar de suportar várias, sendo apenas possível a execução de testes gerais a estas (apenas operações de leitura e escrita sobre a *blockchain*). Nesta fase foram executados os testes e recolhidos e organizados os resultados provenientes destes. Observa-se na tabela 2.9 que esta tarefa começou mais tarde do que o previsto, pelo facto de ser dependente do término da tarefa anterior. A data de fim desta tarefa também sofreu grandes atrasos (dois meses e meio) por ter sido necessário repetir os testes, visto que na primeira tentativa não foram executados corretamente. Foi esta tarefa que levou ao atraso na entrega da dissertação.

Tempo	Data de inicio	Data de fim
Planeado	30/03/2020	01/05/2020
Executado	16/04/2020	17/07/2020

Tabela 2.9: *Benchmarking* de plataformas *blockchain*

Tarefa 5 - Escrita do relatório final

Por fim, a quinta tarefa consistiu na elaboração e escrita do relatório final. Esta começou pela revisão do documento entregue no primeiro semestre e pela escrita do capítulo do estado de arte. À medida que o trabalho ia avançando, foram escritas as partes correspondentes do documento, tais como, o trabalho realizado, a definição dos testes, a sua execução e resultados. A diferença existente na data de início desta tarefa, tal como mostra a tabela 2.10, deve-se à necessidade de revisão de todo o documento e à alteração do foco do trabalho após a reavaliação da escolha da ferramenta a utilizar bem como das plataformas *blockchain* a estudar. O atraso na data de fim (perto de dois meses) deve-se à duração prolongada da tarefa apresentada anteriormente.

Tempo	Data de inicio	Data de fim
Planeado	01/04/2020	29/06/2020
Executado	09/03/2020	08/09/2020

Tabela 2.10: Escrita do relatório final

Esta página foi propositadamente deixada em branco.

Capítulo 3

Background da Tecnologia *Blockchain*

Neste capítulo procede-se a uma introdução à tecnologia *Blockchain*, onde se começa por referir diversos conceitos fundamentais para a sua compreensão, na secção 3.1. De seguida, na secção 3.2, são apresentados os diferentes tipos de *blockchain*. Alguns exemplos de mecanismos de consenso existentes nas *blockchains* privadas, por serem o foco deste trabalho, são apresentados na secção 3.3. Posteriormente, na secção 3.4 é apresentada a definição de *smart contract*. Por fim, na secção 3.6 é feito um breve resumo do capítulo, salientando os aspectos mais importantes deste.

3.1 Conceitos Fundamentais

A tecnologia *Blockchain*, base da *Bitcoin*, foi proposta em 2008, por Satoshi Nakamoto, como uma solução para um sistema de pagamentos eletrónicos livre de entidades provedoras de confiança (Nakamoto, 2008). Este sistema, utilizando os seus mecanismos de verificação e validação, visa resolver o *double spending problem*, que consiste na utilização de um *token* virtual mais do que uma vez, por utilizador (Agrawal, 2017).

O registo de transações é realizado recorrendo ao *ledger* (livro-razão). Com o aparecimento da tecnologia *Blockchain*, novos conceitos de *ledger* surgiram. O *distributed ledger* (livro-razão distribuído), segundo (Brakeville and Perepa, 2019), é um tipo de base de dados compartilhado, replicado e sincronizado entre os membros de uma rede descentralizada, que regista as transações entre os participantes da rede.

Uma *blockchain* corresponde a um *ledger* digital resistente a violações, implementado de forma distribuída - sem um repositório central - e, geralmente, sem uma autoridade central. Ela permite que os utilizadores de uma comunidade registem transações num livro partilhado dentro desta, sendo que nenhuma transação pode ser alterada depois de publicada (Yaga et al., 2018). Estes utilizadores mantêm uma cópia do *ledger*, aplicando transações que foram validadas por um protocolo de consenso. Estas são agrupadas em blocos que incluem um *hash* - que garante a integridade da informação, garantindo que não sofre alterações, quer propositadamente quer acidentalmente - que liga cada bloco ao imediatamente anterior (The Linux Foundation, 2018).

De seguida são apresentados os componentes constituintes de uma *blockchain*. É de salientar que a constituição de cada *blockchain* e dos seus blocos varia conforme a im-

plementação desta. Os componentes que são referidos de seguida correspondem então a uma implementação genérica de uma *blockchain*. Na figura 3.1 encontra-se representada a *blockchain* B, constituída pelos blocos B0, B1, B2 e B3. O bloco B0 diz respeito ao *genesis block*. Este é o bloco inicial de uma *blockchain* e, por esta razão, não contém transações nem bloco precedente (The Linux Foundation, 019b). Os blocos de uma *blockchain* são interligados através do armazenamento do *hash* das suas transações e uma cópia do *hash* do bloco anterior no seu *header*. Assim, é garantida a imutabilidade pois qualquer alteração feita num bloco é fácil de detectar, comparando o *hash* do bloco original com o *hash* do bloco alterado (Yaga et al., 2018). As transações que ocorrem na *blockchain* são agrupadas em blocos.

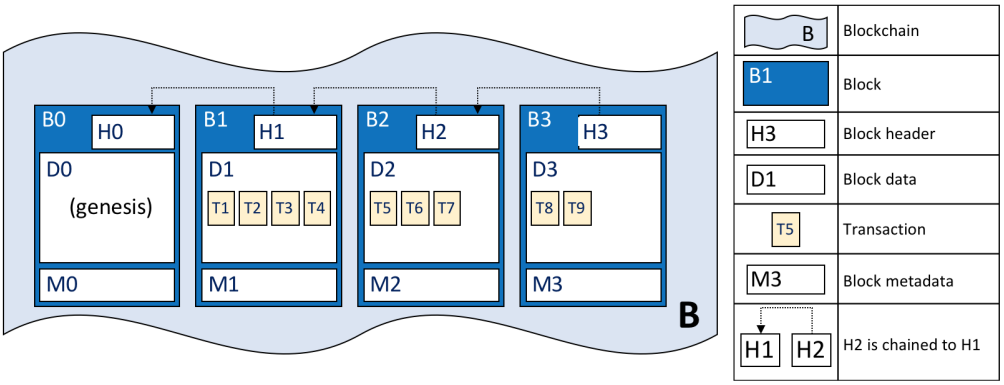


Figura 3.1: Estrutura da *blockchain*, (The Linux Foundation, 019b)

Um bloco encontra-se dividido em três partes: o *block header*, o *block data* e o *block metadata*. Na figura 3.2 é apresentada a composição de um bloco.

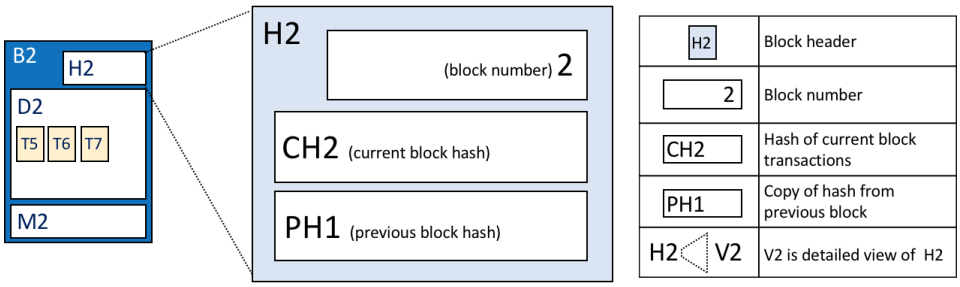


Figura 3.2: Bloco da *blockchain*, (The Linux Foundation, 019b)

O *block header* (H2), tal como mencionado acima, contém a *hash* do bloco atual (CH2) e a *hash* do bloco anterior (PH1). Para além disto, possui um *block number*, definido por um inteiro, que começa no 0 e que vai incrementando à medida que vão sendo adicionados novos blocos à *blockchain*. No *block data* está contida a lista de transações, inseridas de forma sequencial (T5, T6, T7). O *block metadata* contém informações acerca da criação do bloco e do seu criador (The Linux Foundation, 019b).

3.2 Tipos de *Blockchains*

As *Blockchains* podem ser categorizadas com base no seu modelo de permissão (Yaga et al., 2018). Este modelo está diretamente relacionado com a possibilidade de determinar quem tem autorização para participar na rede, executar o protocolo de consenso implementado na *Blockchain* e manter o registo atualizado do *ledger* (Jayachandran, 2017). Assim, são introduzidos os conceitos de *blockchain* pública, *blockchain* privada, *blockchain* de consórcio e *blockchain* híbrida, para melhor entender as diferenças entre estas.

Numa ***blockchain* pública** é permitida a entrada de qualquer utilizador na rede. Estes utilizadores mantêm uma cópia atualizada do *ledger*, conseguindo assim executar os mecanismos de consenso presentes na rede e enviar e/ou receber transações de outros utilizadores (Yaga et al., 2018). Todas as transações que ocorrem são totalmente transparentes, isto significa que qualquer pessoa pode aceder aos seus detalhes (Dragonchain, 2019). Os mecanismos de consenso decidem qual o bloco a ser inserido de seguida na *blockchain* (Zheng et al., 2017), alguns destes serão detalhados na secção 3.3.

Neste tipo de *blockchain* não existe forma de associar o endereço público dos utilizadores a uma identidade real, permanecendo assim anónimos (Buterin, 2015). O anonimato dos utilizadores torna difícil o acesso a estes caso seja necessário rastreá-los ou expulsá-los da rede, por apresentarem um comportamento incorreto na mesma (Dragonchain, 2019).

Estas *blockchains* são desenhadas para serem totalmente descentralizadas, sem que nenhum indivíduo ou entidade tenha controlo sobre quais as transações que devem ser registadas na *blockchain* ou a ordem pela qual devem ser processadas (Dragonchain, 2019).

Uma ***blockchain* privada** apresenta um modelo em que a entrada dos utilizadores na rede está dependente de um convite por parte de uma autoridade responsável pelas permissões (Jayachandran, 2017). Ao ser conhecida a identidade dos participantes, o nível de confiança entre estes aumenta, aumentando assim o controlo na rede, uma vez que sobre um utilizador passa a ser possível a aplicação de restrições de leitura e escrita na *blockchain* (Yaga et al., 2018). Por todos os motivos mencionados, a introdução de informação incorreta na *blockchain* é reduzida, pois a assinatura digital de cada utilizador acompanha todas as suas operações realizadas na rede, sendo mais fácil a possibilidade de este ser rastreado e expulso da mesma (The Linux Foundation, 2018).

Estas *blockchains* garantem uma certa confidencialidade pois, apesar de existir um registo das transações efetuadas entre dois utilizadores, o conteúdo dessas transações apenas pode ser acedido pelas partes envolvidas (Yaga et al., 2018).

As ***blockchains* de consórcio** são consideradas uma designação separada das *blockchains* privadas. A principal diferença entre estas é o facto de as cadeias de blocos de consórcio serem governadas por um grupo e não apenas por uma única entidade. Estas *blockchains* podem, por este motivo, ser consideradas uma subcategoria das *blockchains* privadas (Dragonchain, 2019).

A ***blockchain* híbrida** combina os benefícios de privacidade existentes numa *blockchain* privada com os de segurança e transparência existentes nas *blockchains* públicas. Esta combinação oferece às empresas uma grande flexibilidade, pois pode escolher quais os dados que quer tornar públicos e transparentes, e quais prefere manter em sigilo (Dragonchain, 2019).

Apresentados os diversos tipos de *blockchains* existentes, é importante realçar que o foco principal deste trabalho são plataformas pertencentes ao domínio das *blockchains* privadas e de consórcio.

3.3 Mecanismos de Consenso das *Blockchains* Privadas

Determinar qual o próximo utilizador a publicar um bloco na rede é um aspeto fundamental da tecnologia *blockchain*. Tal decisão é conseguida recorrendo aos mecanismos de consenso (Yaga et al., 2018).

Na *blockchain*, como chegar a um consenso entre os nós que não confiam uns nos outros é uma adaptação do Problema dos Generais Bizantinos (Lamport et al., 1982). Neste problema, um grupo de generais e o seu exército bizantino cercam uma cidade. Alguns desses querem atacar, enquanto outros preferem recuar. No entanto, o ataque falharia se apenas uma parte destes atacasse a cidade. Assim, estes precisam de chegar a um acordo, mesmo existindo um problema de confiança (Lamport et al., 1982).

Chegar a um consenso num ambiente distribuído é um desafio, tal como acontece numa *blockchain*, tratando-se de uma rede distribuída, por não existir um nó central que garanta que os *ledgers* dos nós sejam todos iguais. Apesar de não existir essa autoridade central, é necessário que os nós concordem na validação das transações. Por este motivo, são implementados protocolos nesta que assegurem esta conformidade, de forma a obter consenso numa *blockchain* (Zheng et al., 2017). Os mecanismos de consenso garantem que as regras implementadas na rede são seguidas e que todas as transações nela existentes ocorrem de forma correta e confiável.

Nesta secção procede-se então à apresentação de dois modelos de mecanismos de consenso implementados em *blockchains* privadas, mais precisamente nas plataformas *Hyperledger Fabric*, *Quorum* e FISCO BCOS, por serem o principal foco deste trabalho, sendo eles:

- *Raft*;
- *Istanbul Byzantine Fault Tolerance*.

Estes mecanismos de consenso são apresentados em detalhe nas subsecções que se seguem.

Os mecanismos de consenso implementados em *blockchains* públicas, bem como os restantes implementados em *blockchains* privadas, podem ser consultados nos trabalhos de (Yaga et al., 2018), (Zheng et al., 2017) e (Frankenfield, 2019).

3.3.1 *Raft*

Este mecanismo de consenso é utilizado pelas *blockchains Hyperledger Fabric*, *Quorum* e FISCO BCOS, que são alvo de estudo neste trabalho. O *Raft* surgiu de uma adaptação do *Paxos* (Ongaro and Ousterhout, 2014). Devido às limitações do *Paxos* - como a sua difícil compreensão e implementação - o *Raft* apresentou-se como uma forma de as resolver, assegurando que as caraterísticas de performance e tolerância a falhas permanecessem intactas (Huang et al., 2018).

O funcionamento do *Raft* é resumido, de uma forma geral, na figura 3.3. O servidor é constituído pela *state machine*, pelo *log* e pelo *consensus module*. O cliente efetua um pedido a um servidor. Ao receber esse pedido, o *consensus module* insere-o no seu *log* e, ao mesmo tempo, transmite-o para os *consensus modules* dos outros servidores. Desta forma, é possível assegurar as mesmas informações em todos estes. Após todos estes passos

e corrigidos eventuais problemas, os pedidos guardados no *log* são enviados para a *state machine*, para que proceda à sua execução. O resultado proveniente desta é posteriormente enviado para o cliente.

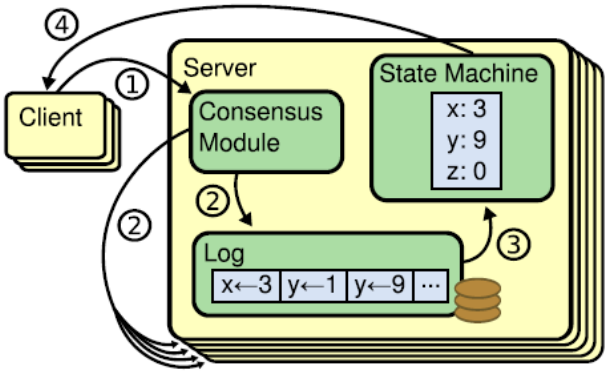


Figura 3.3: Funcionamento do *Raft*, (Ongaro and Ousterhout, 2014)

Neste mecanismo de consenso, os servidores podem assumir três estados distintos na rede *blockchain*:

- **Líder** - neste estado, o servidor é responsável por responder a todos os pedidos dos clientes. Existe apenas um líder na rede (Ismail and Materwala, 2019);
- ***Follower* (Seguidor)** - o servidor comporta-se de forma passiva e simplesmente responde às solicitações feitas por parte do líder e dos candidatos existentes na rede (Ismail and Materwala, 2019);
- **Candidato** - os servidores que se encontram neste estado assumem a forma de um candidato à eleição, mas apenas durante o período de tempo em que esta ocorre (Ismail and Materwala, 2019).

Todos os servidores são iniciados no estado seguidor. As transições de estados pelas quais um servidor pode passar são demonstradas na figura 3.4 e explicado o funcionamento deste fluxo logo de seguida.

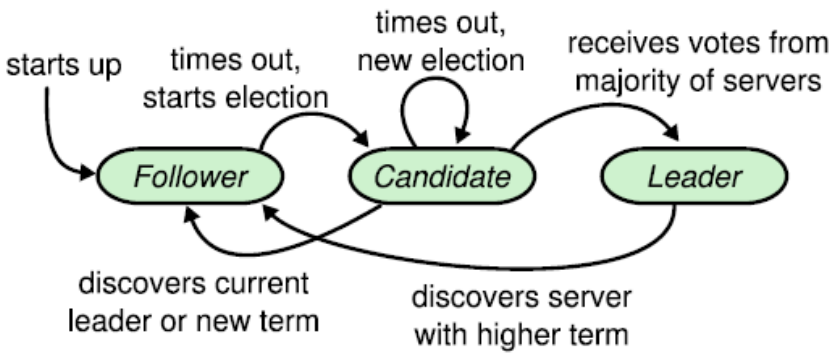


Figura 3.4: Estados de um Servidor, (Ongaro and Ousterhout, 2014)

Devido à existência de um líder na rede, o funcionamento do *Raft* pode ser dividido em etapas:

1. **Eleição do Líder** - o tempo de execução é divido em partes de comprimento arbitrário denominadas por *terms*. Um *term* é iniciado com uma eleição, onde o líder é responsável por operar durante o período de duração do mesmo. Se este terminar sem que seja eleito um líder, é iniciado outro com uma nova eleição (Ongaro and Ousterhout, 2014). Esta divisão do tempo em *terms* é ilustrada na figura 3.5.

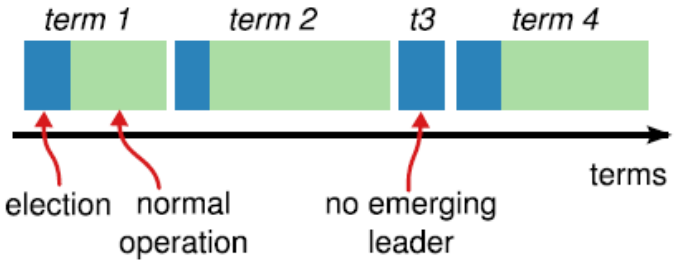


Figura 3.5: Tempo dividido em *terms*, (Ongaro and Ousterhout, 2014)

A comunicação entre os servidores é feita com recurso a Remote Procedure Calls (RPCs), sendo que estes necessitam apenas de duas chamadas durante o seu processo de execução - a *RequestVote* e a *AppendEntries*. A primeira, *RequestVote* RPCs, é executada apenas pelos candidatos, sendo que a sua função é reunir votos de outros servidores durante o período de uma eleição.

No caso em que um seguidor deixa de receber comunicações do seu líder (os chamados *heartbeats*), uma nova eleição é iniciada, após atingir o *election timeout*. Este período de tempo diz respeito ao tempo necessário que um seguidor tem para se tornar candidato, assumindo que não existe nenhum líder nesse momento. Ao iniciar uma eleição, um seguidor incrementa o seu índice e atualiza o seu estado para candidato. De seguida, o candidato vota nele mesmo e envia *RequestVote* RPCs para os restantes servidores. Deste processo podem resultar três possíveis resultados:

- Se o candidato obtiver a maioria dos votos, este vence a eleição e envia uma mensagem a todos os outros, declarando-se como líder (Ismail and Materwala, 2019).
- Enquanto aguarda a votação, um candidato pode receber uma mensagem de autoridade de outro servidor que afirma ser um líder. O candidato compara o valor do índice do *term* atual com o *term* do servidor que se está a afirmar como líder. Se o índice do *term* do servidor for inferior ao do candidato, o candidato rejeita essa mensagem. Se o índice for superior ao do candidato, o candidato reconhece o servidor como líder e retorna ao estado de seguidor (Ismail and Materwala, 2019).
- Uma eleição pode acabar sem que seja eleito um novo líder. Isto acontece caso muitos seguidores iniciem o processo eleitoral ao mesmo tempo. Desta forma, os votos podem ser divididos de modo a que nenhum candidato obtenha a maioria dos votos (Ismail and Materwala, 2019). Esta situação é rara, uma vez que a implementação dos tempos de *election timeout* é feita de forma aleatória. No entanto, caso aconteça, todos os servidores dão *timeout* e de seguida é iniciada uma nova eleição (Dias, 2019).

2. **Replicação de *Logs*** - Momento em que ocorre a segunda chamada, *AppendEntries*, sendo realizada apenas pelos líderes. Esta tem como função a replicação dos pedidos recebidos para todos os *consensus modules* dos restantes servidores, logo após ser inserido, em primeiro lugar, no *log* do *consensus module* que recebeu o pedido. Desta forma, é assegurada a sincronização entre todos os servidores (Ongaro and Ousterhout, 2014).

O Raft resolve o problema da tolerância a falhas, mas não fornece integridade dos dados no caso em que um nó se comporta de forma incorreta. Por exemplo, se o líder se comportar desta maneira e executar transações inválidas, toda a rede *blockchain* se tornará inválida (Ismail and Materwala, 2019).

3.3.2 *Istanbul Byzantine Fault Tolerance*

O Istanbul Byzantine Fault Tolerance (IBFT) é um mecanismo de consenso que consiste numa alternativa ao Proof-of-Work (PoW). De forma semelhante a outros algoritmos, o IBFT garante uma ordem única e acordada para transações na *blockchain*, fornecendo benefícios adicionais para as empresas (PegaSys, 2018). O IBFT está implementado na *Quorum*, uma das plataformas que é alvo de estudo neste trabalho.

Neste mecanismo de consenso podemos encontrar dois tipos de nós: os nós validadores (referidos como autoridades quando se encontram vinculados a entidades físicas) e os nós regulares. Os nós de autoridade são os principais responsáveis pela criação dos blocos (Prusty, 2018).

Este algoritmo apresenta as seguintes caraterísticas, explicadas em (PegaSys, 2018):

- **Tempo reduzido para construção e validação de blocos** - O esforço necessário para construir e validar os blocos é significativamente reduzido, aumentando assim o rendimento da cadeia.
- **Alta Integridade dos dados e Tolerância a Falhas** - O IBFT utiliza um grupo de validadores para garantir a integridade de cada bloco proposto. É necessária a aprovação de cerca de 66% destes validadores para o assinar antes de o inserir na cadeia, tornando muito difícil a falsificação do mesmo. A "liderança"do grupo também vai alternando ao longo do tempo, assegurando assim que um nó malicioso ou defeituoso não possa exercer influência a longo prazo na cadeia.
- **Operacionalmente Flexível** - O grupo de validadores pode ser modificado, assegurando que o grupo contém apenas nós de confiança total.

No IBFT, à semelhança do que acontece no algoritmo *Raft*, os nós também assumem diferentes estados na rede *blockchain*:

- *Awaiting Proposal* - O nó validador está à espera que um novo bloco seja fornecido pelo *proposer* atual. Se o validador for o *proposer* para esta ronda, este prepara o bloco proposto e transmite-o através de uma mensagem de pré-preparação (PegaSys, 2018).
- *Preparing* - Um bloco válido foi recebido e os nós validadores são notificados. Entretanto, aguarda-se a notificação de aceitação por parte de todos os nós validadores (PegaSys, 2018).

- *Ready* - Se todos os nós validadores aceitaram o bloco, então este está pronto para ser inserido na *blockchain* (PegaSys, 2018).
- *Round Change* - Ou a ronda atingiu o tempo limite (*timeout*) antes de ser alcançado o consenso de todos os nós ou o bloco não foi inserido com sucesso. Nestas circunstâncias, é necessário que todos os validadores concordem em começar uma nova ronda (PegaSys, 2018).

A figura 3.6 ilustra os estados pelos quais os nós passam, bem como as respetivas transações.

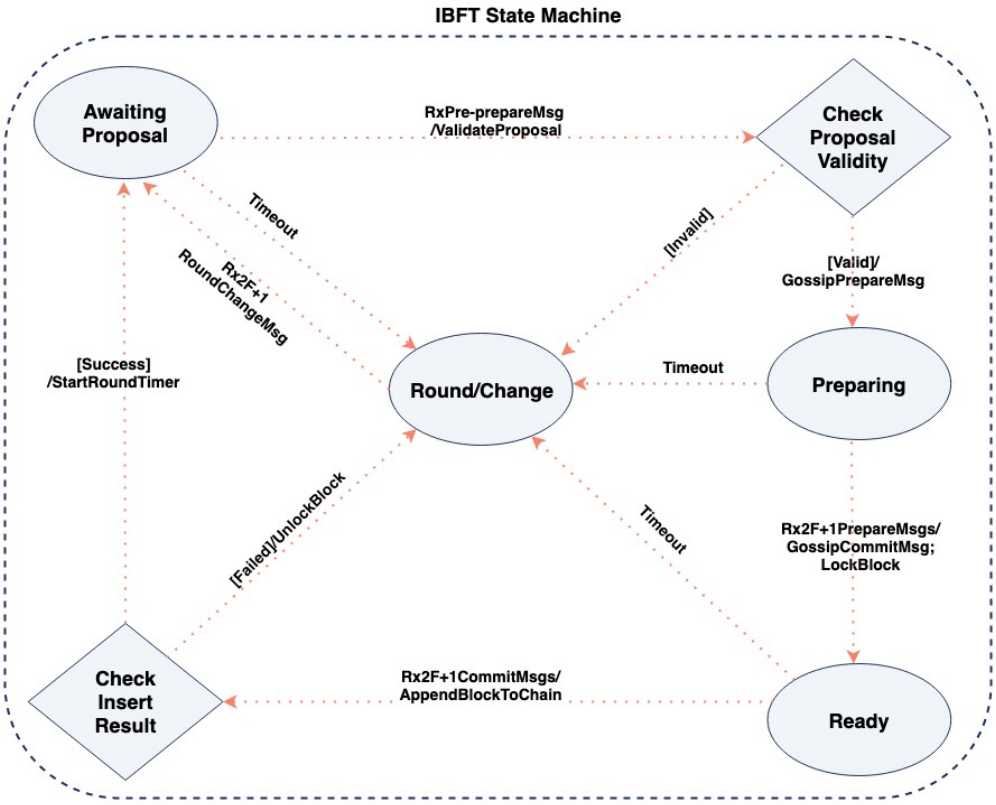


Figura 3.6: Estados de um nó, (Cheah et al., 2018)

3.3.3 Análise Geral

As principais diferenças entre os mecanismos de consenso descritos nas secções 3.3.1 e 3.3.2 são apresentadas na tabela 3.1, adaptada de (Nolan, 2018). Pode observar-se que estes podem ser divididos quanto ao tipo de consenso que implementam: Crash Fault Tolerance (CFT) ou Byzantine Fault Tolerance (BFT). A CFT protege apenas contra a falha dos nós, enquanto a BFT protege a *blockchain* contra utilizadores maliciosos.

No *Raft* não existe qualquer proteção contra utilizadores maliciosos. As informações do bloco podem ser modificadas e o *hash* correspondente alterado. Uma alternativa passaria por implementar uma solução personalizada que armazenaria os *hashes* dos blocos num sistema à parte. No IBFT, torna-se difícil de adulterar o passado, pois o utilizador precisaria de todas as chaves de assinatura privadas dos nós validadores para editar os blocos. Contudo, assim como em todos os algoritmos de votação, neste mecanismo de consenso existe um *overhead* de mensagens que piora exponencialmente à medida que são adicionados mais nós validadores à rede (Nolan, 2018).

Mecanismo de Consenso	**Tipo de Consenso**	**Vantagens**	**Desvantagens**
Raft	CFT	- Rápido na cunhagem do bloco; - Inexistência de *forks*.	- Nenhuma proteção contra utilizadores maliciosos;
IBFT	BFT	- Proteção contra utilizados maliciosos; - Difícil de alterar os blocos passados.	- *Overhead* de mensagens.

Tabela 3.1: Comparação entre *Raft* e *IBFT*, adaptada de (Nolan, 2018)

3.4 *Smart Contracts*

O conceito *smart contract* foi definido pela primeira vez por (Szabo, 1997) como:

"*a computerized transaction protocol that executes the terms of a contract*".

Os seus principais objetivos passam por satisfazer as condições comuns de um contrato, minimizar atividades maliciosas e diminuir a necessidade de intermediários de confiança (Yaga et al., 2018).

Com o avanço da tecnologia *blockchain*, também os *smart contracts* evoluíram, passando a ser definidos como:

"a collection of code and data (...) that is deployed using cryptographically signed transactions on the blockchain network" (Yaga et al., 2018).

Contudo, é de referir que nem todas as blockchains conseguem armazenar e/ou executar *smart contracts* (Yaga et al., 2018), como é o caso da *blockchain Bitcoin* (Dias, 2019).

Por estarem contidos na *blockchain*, os *smart contracts* são visíveis e executados por todos os participantes desta, que, assim, podem analisar o seu código e os resultados da sua execução (Christidis and Devetsikiotis, 2016).

De forma a não tornar o documento extenso, as diversas aplicações de um *smart contract* e exemplos do mesmo, podem ser consultados no documento (Dias, 2019).

3.5 Plataformas *Blockchain*

Nesta secção são apresentadas as plataformas *blockchain* que são alvo de estudo neste trabalho (*Hyperledger Fabric* e FISCO BCOS) e a plataforma *Quorum*, que não foi possível estudar. Assim, inicia-se com a apresentação da plataforma *Hyperledger Fabric*, na secção 3.5.1. De seguida, a explicação da plataforma *Quorum*, na secção 3.5.2 e, por fim, a exposição da FISCO BCOS, na secção 3.5.3. Em cada uma das secções é explicado o funcionamento de cada uma delas bem como a sua arquitetura.

3.5.1 Hyperledger Fabric

A *Hyperledger Fabric* é uma plataforma de código aberto, projetada para uso em contextos empresariais (The Linux Foundation, 2018). Esta plataforma apresenta uma arquitetura altamente modular e configurável, permitindo assim inovação, versatilidade e otimização para uma ampla gama de casos de uso da indústria.

A *Fabric* é a primeira a oferecer suporte para *smart contracts* criados em linguagens de programação de uso geral (*Java*, *Go* e *NodeJS*), em vez de linguagens específicas de domínio restrito (*Solidity*). Isto consiste numa mais valia para as empresas, visto que, desta forma, possuem o conjunto de capacidades necessárias para os desenvolver, não sendo necessária nenhuma espécie de formação (The Linux Foundation, 2018).

A *Hyperledger* é uma *blockchain* privada, em que os participantes na rede são conhecidos entre si. Assim, embora não exista confiança total entre estes (podendo tratar-se de concorrentes no mesmo setor, por exemplo), a rede pode ser operada sob um modelo de governança construído com base na confiança entre os utilizadores - tal como um acordo para lidar com as disputas (The Linux Foundation, 2018).

A *Fabric* é considerada uma das plataformas com melhor desempenho disponível atualmente, tanto em termos de processamento de transações quanto na latência de confirmação destas (The Linux Foundation, 2018). Permite privacidade e confidencialidade das transações e dos *smart contracts*, denominados *chaincodes* no contexto desta plataforma (The Linux Foundation, 2018).

Relativamente à arquitetura desta plataforma, esta segue o modelo ***execute-order-validate***. De forma breve, este consiste no envio de propostas de transações por parte de um cliente para os *peer nodes* que executam o *chaincode* correspondente. Após a sua execução esta é reenviada para o cliente sob a forma de *proposal response*, que posteriormente é enviada para o serviço de ordenamento de transações. Depois de serem ordenadas todas as transações recebidas, estas são agrupadas em blocos que passam por um processo de validação antes de este serem adicionados à *blockchain*. Todo este processo encontra-se ilustrado na figura 3.7, retirada de (Sukhwani et al., 2018).

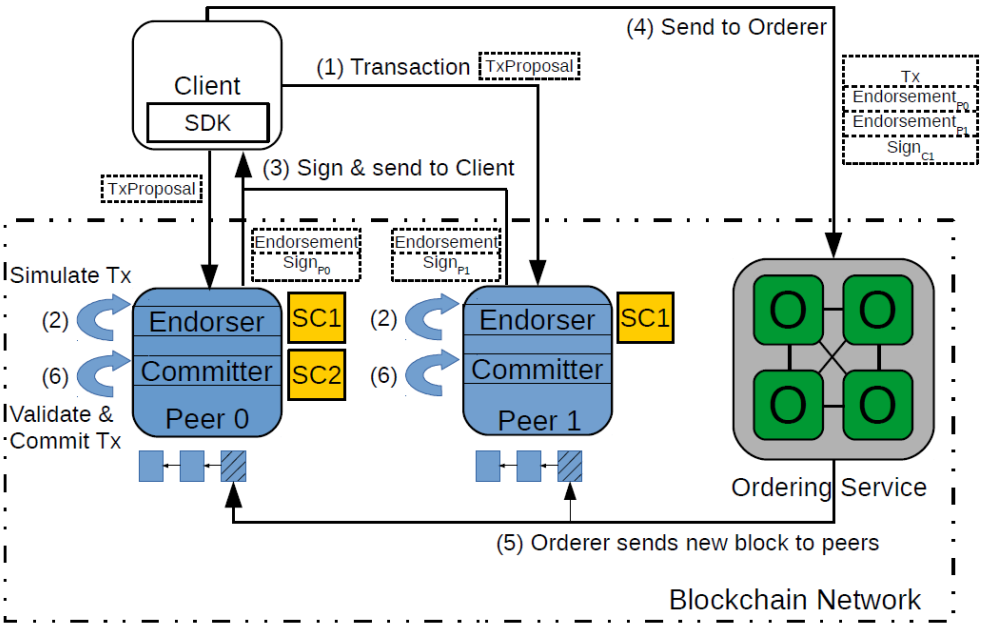


Figura 3.7: Arquitetura da *Hyperledger Fabric*, retirada de (Sukhwani et al., 2018)

3.5.2 Quorum

A plataforma *Quorum* é uma *blockchain* baseada na *Ethereum*, que foi desenvolvida para trabalhar com serviços financeiros, mas que pode ser estendida a qualquer indústria que queira fazer uso desta (JPMorgan Chase & Co., 2016).

O setor financeiro exige um tipo específico de *blockchain*, devendo esta ser rápida e funcionar enquanto mantém sob controlo a privacidade dos participantes na rede (Lamounier, 2019). A plataforma *Quorum* foi desenvolvida por JP Morgan exatamente com este propósito, suportando privacidade nas transações e contratos da *blockchain* (JPMorgan Chase & Co., 2016). A *Quorum* tem como base o código *Ethereum* e, por esta razão, herda assim as suas caraterísticas, apresentando algumas extensões a esse código, tais como:

- A privacidade nas transações e contratos da blockchain;
- Mecanismos de consenso baseados em múltiplas rondas de votos;
- Uma gestão de permissões da rede blockchain;
- Uma melhor performance.

De forma a conseguir privacidade nas transações e nos contratos presentes na *blockchain*, a plataforma *Quorum* passou a suportar dois estados: o estado público, acessível a todos os nós participantes da rede; e o estado privado, em que apenas os nós com as permissões corretas podem aceder. Assim, todos os nós participantes na rede partilham o estado público, mantendo na sua posse o estado privado (JPMorgan Chase & Co., 2016).

A figura 3.8, retirada de (JPMorgan Chase & Co., 2016), representa a arquitetura da plataforma *Quorum* bem como a constituição dos seus componentes, sendo estes posteriormente detalhados e explicados.

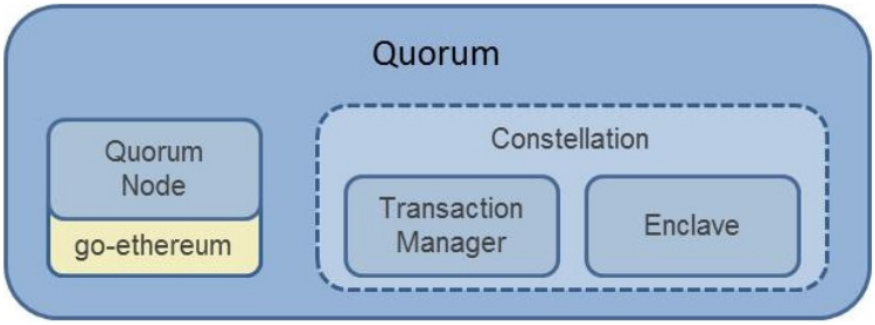


Figura 3.8: Arquitetura da *Quorum*, (JPMorgan Chase & Co., 2016)

A constituição da arquitetura da plataforma *Quorum* é composta por duas grandes partes: o *Quorum Node* e o *Constellation*. Sendo que este último se encontra dividido em dois componentes - o *Transaction Manager* e o *Enclave*.

O primeiro componente, o ***Quorum Node***, diz respeito a um *fork* da rede *Ethereum*, de forma a aproveitar as vantagens desta (JPMorgan Chase & Co., 2016). No entanto, foram realizadas diversas modificações a este para que se adaptasse à *blockchain Quorum*, referidas em (JPMorgan Chase & Co., 2016).

Outro componente da arquitetura *blockchain* da *Quorum* é o ***Constellation*** que, como já foi referido acima, engloba o *Transaction Manager* e o *Enclave*. Este consiste num sistema que garante que as informações adicionadas à *blockchain* permanecem seguras (Lamounier, 2019).

O ***Transaction Manager*** é responsável pela privacidade das transações, permitindo a troca de operações que contêm dados encriptados sem que se aceda às chaves privadas responsáveis por essa encriptação (JPMorgan Chase & Co., 2016).

Na plataforma *Quorum*, uma transação pode ser do tipo pública ou privada. Para identificar o seu tipo, foi-lhe adicionado o campo *privateFor* - que, se apresentar a chave pública do recetor desta, então é do tipo privada (JPMorgan Chase & Co., 2016). A criptografia garante que os dados são mantidos em segurança durante todo o processo (Lamounier, 2019).

Por fim, o ***Enclave*** é o responsável por facilitar diferentes técnicas criptográficas, tais como a autenticação dos participantes e o histórico de transações. O *Transaction Manager* delega o trabalho de criptografia/descriptografia para o *Enclave* (Lamounier, 2019).

3.5.3 FISCO BCOS

A FISCO BCOS é uma plataforma *blockchain* de consórcio, de código aberto. O seu desempenho já atingiu valores superiores a 10000 transações por segundo (BCOS, 2020). A plataforma fornece vários recursos, entre eles: protocolos de comunicação, mecanismos de consenso, algoritmos de proteção de privacidade, algoritmos de criptografia e armazenamento distribuído (BCOS, 2020).

Na versão 2.0, a FISCO BCOS propôs uma arquitetura de "*one-body, two-wing, multi-engine*" (um corpo, duas asas e vários motores) para expandir os valores de taxa de transferência do sistema e melhorar significativamente o seu desempenho (BCOS, 2020). Este

modelo encontra-se representado na figura 3.9.

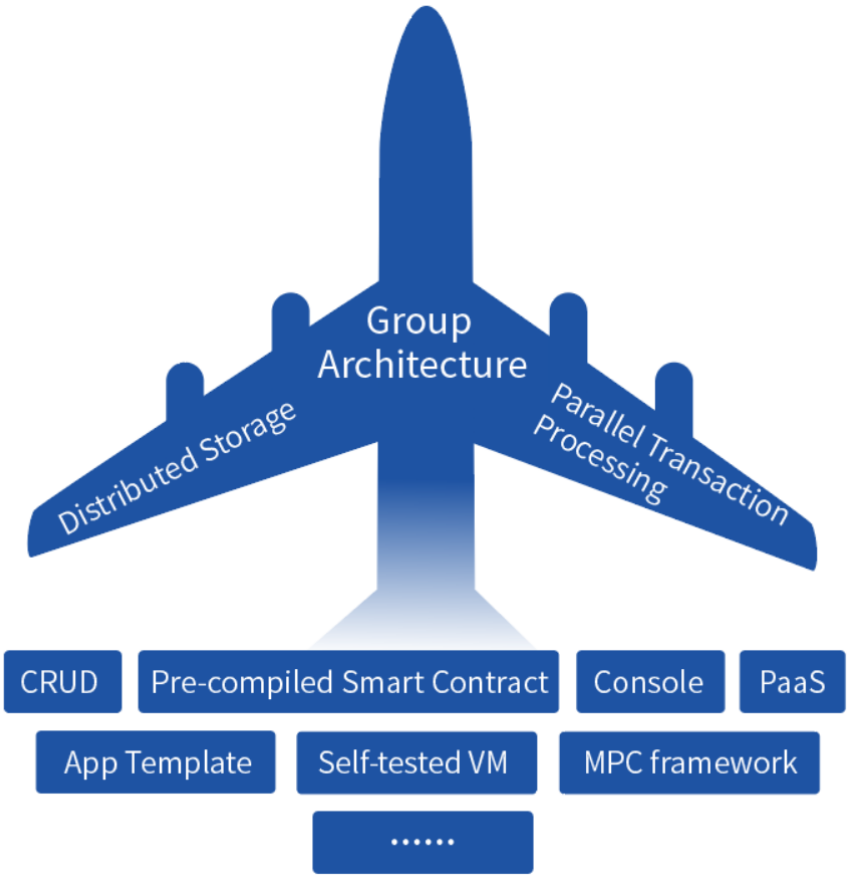


Figura 3.9: Arquitetura da FISCO-BCOS, (BCOS, 2020)

O "*one-body*"refere-se à estrutura do grupo, suporta a rápida formação de uma *blockchain* de consórcio e permite que as empresas comuniquem facilmente. As empresas podem escolher grupos diferentes para partilhar dados e obter consenso de vários *ledgers* diferentes (BCOS, 2020).

As "*two-wing*"referem-se ao suporte de modelos de computação paralela e armazenamento distribuído, os quais proporcionam melhor escalabilidade ao grupo de arquitetura (BCOS, 2020).

O "*multi-engine*"é um resumo de uma série de recursos funcionais (BCOS, 2020).

Esta plataforma *blockchain* suporta o mecanismo de consenso *Raft* e as bases de dados *LevelDB* e *MySQL* (BCOS, 2020).

3.6 Resumo do Capítulo

Neste capítulo foi feita uma breve introdução teórica à tecnologia *blockchain*, começando pela definição e composição da mesma.

De seguida, foram apresentados os diferentes tipos de *blockchain*: pública, privada, de consórcio e híbrida. Tendo em especial atenção as *blockchains* privadas por serem alvo de estudo neste trabalho. Os mecanismos de consenso implementados pelas plataformas *blockchain* escolhidas, foram também referidos neste capítulo. Estes são fulcrais no funcionamento destes sistemas, visto que são os responsáveis pela validação e aprovação das transações.

Por fim, foram referidas as plataformas *blockchain* que se pretende estudar: a *Hyperledger Fabric* e a FISCO BCOS. É também mencionada a *Quorum*, apesar de esta não ter sido analisada neste trabalho.

Esta página foi propositadamente deixada em branco.

Capítulo 4

Estado da Arte de *Benchmarks* de Plataformas *Blockchain*

Neste capítulo procede-se à exposição de trabalhos relacionados de *benchmarking* de plataformas *blockchain*. Em primeiro lugar, na secção 4.1 são apresentados os trabalhos realizados por diversos autores. De seguida, na secção 4.3, é feita a comparação entre as diversas ferramentas encontradas para a elaboração de testes de *benchmark* de plataformas *blockchain*. Por fim, na secção 4.4, são apresentadas as conclusões relativas à análise das ferramentas.

4.1 *Benchmarks* Existentes

Com a evolução da tecnologia *blockchain*, tornou-se fulcral perceber o seu funcionamento e avaliar o seu nível de performance. Com este propósito, diversos autores realizaram trabalhos que pretendiam testar o seu desempenho.

Houve quem implementasse os seus próprios cenários de teste, enquanto outros recorreram a ferramentas já existentes para a sua elaboração. De seguida, são apresentados alguns trabalhos realizados nesta área.

Em (Pongnumkul et al., 2017) foi realizado o estudo de duas plataformas *blockchain*: a *Ethereum* e a *Hyperledger Fabric*. Neste estudo, foi avaliado o desempenho das plataformas quando estas são submetidas a um número elevado de transações invocadas pelos utilizadores. Para este fim foi criada uma aplicação onde era possível simular estas transferências. Os autores não tiveram em consideração os mecanismos de consenso aquando da realização das suas experiências, por serem utilizados mecanismos diferentes em cada uma das plataformas.

Em (Thakkar et al., 2018), foi analisada a plataforma *Hyperledger Fabric*, na versão 1.0. Neste trabalho foram variados diferentes parâmetros de configuração, tais como a base de dados adotada, o tamanho dos blocos, o consumo de recursos e a política de aprovação de uma transação. Variando estes aspetos, foi possível estudar o desempenho da plataforma *Hyperledger Fabric*, quanto à sua taxa de transferência (*throughput*) e latência.

É importante realçar que as sugestões fornecidas por (Thakkar et al., 2018), para otimizar esta plataforma, foram adotadas posteriormente nas versões seguintes da *Hyperledger Fabric*.

Nas subsecções que se seguem são apresentadas as ferramentas de *benchmark* encontradas bem como trabalhos que as utilizaram.

4.2 Ferramentas de *Benchmark*

Nesta secção são apresentadas três ferramentas de *benchmarking* de plataformas *blockchain*: em primeiro lugar, na subsecção 4.2.1, a *BLOCKBENCH*, posteriormente, na subsecção 4.2.2, a *Hyperledger Caliper* e, por fim, na subsecção 4.2.3, a *Gauge*. Em cada uma das subsecções é referido o seu funcionamento bem como as suas caraterísticas principais.

4.2.1 BLOCKBENCH

A ferramenta BLOCKBENCH é descrita como a primeira a proporcionar o *benchmarking* de plataformas privadas, de forma a ser possível o seu estudo e comparação (Dinh et al., 2017).

A BLOCKBENCH avalia o desempenho geral das plataformas *blockchain*, tendo em consideração componentes como a taxa de transferência, a latência, a escalabilidade e a tolerância a falhas. Também é possível realizar uma análise relativa à segurança das plataformas, sendo para isso simulados ataques ao nível da rede.

Neste trabalho foi utilizada a ferramenta BLOCKBENCH para realizar uma avaliação abrangente de três plataformas blockchain: *Ethereum*, *Parity* e *Hyperledger Fabric*. Os resultados a que estes autores chegaram permitiu-lhes afirmar que estes sistemas apresentam ainda um número de transações relativamente baixo quando comparados com os que são alcançados pelas base de dados atuais, apresentando uma performance inferior.

A arquitetura desta ferramenta encontra-se representada na figura 4.1. Os autores salientam os componentes:

- *IWorkloadConnector* - que permite a adição de novos *workloads* para teste;
- *Driver* - que permite a recolha de informação, contida no *StatsColletor*, relativamente aos testes executados na *blockchain*. As configurações destes testes estão presentes no *Configuration*;
- *IBlockchainConnector* - que permite a adição de novas plataformas *blockchain* para a realização de testes.

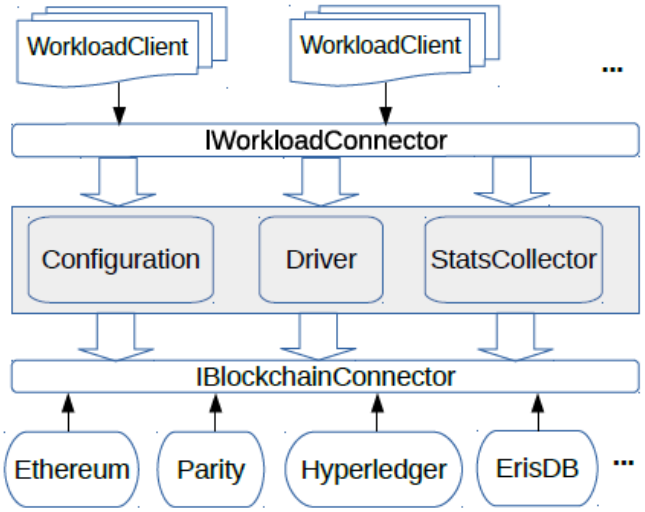


Figura 4.1: Arquitetura da ferramenta BLOCKBENCH, (Dinh et al., 2017)

No que diz respeito à constituição de uma *blockchain*, os autores consideram que esta é composta por quatro camadas de abstração: a camada de consenso, a camada de modelo de dados, a camada de execução e a camada de aplicação. Esta divisão encontra-se ilustrada na figura 4.2.

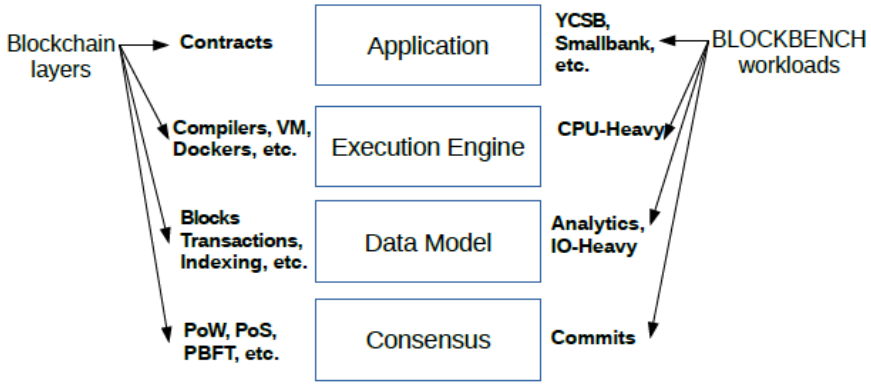


Figura 4.2: Camadas de abstração de uma *blockchain*, (Dinh et al., 2017)

Os *workloads* implementados pelos autores foram divididos em duas categorias:

- *Macro Benchmarks* - para avaliar a performance da camada de aplicação;
- *Micro Benchmarks* - para testar as restantes camadas.

Os autores implementaram ainda um *smart contract* para cada *workload*. Para as plataformas *Parity* e *Ethereum*, este foi implementado em *Solidity*, enquanto que para a *Hyperledger Fabric*, foi usada *Golang*.

4.2.2 Hyperledger Caliper

A *Hyperledger Caliper* é descrita em (The Linux Foundation, 2019). Trata-se de uma ferramenta que permite avaliar o desempenho de diversas *blockchains*, com casos de uso personalizados.

Atualmente, a *Caliper* suporta diversas soluções *blockchain*, sendo elas (The Linux Foundation, 2019):

- *Hyperledger Besu*
- *Hyperledger Burrow*
- *Hyperledger Fabric*
- *Hyperledger Iroha*
- *Hyperledger Sawtooth*
- *Ethereum*
- FISCO BCOS

Utilizando esta ferramenta é possível recolher métricas que dizem respeito à taxa de transferência e latência de operações de leitura e escrita, à taxa de sucesso de transações e ao consumo de recursos por parte dos nós que constituem a rede.

Na figura 4.3 encontra-se representada a arquitetura da *Caliper*. De uma forma geral, esta gera um *workload* sobre a plataforma a testar (System Under Test (SUT)) e monitoriza continuamente as suas respostas. No final, é gerado um relatório com base nas informações recolhidas do SUT.

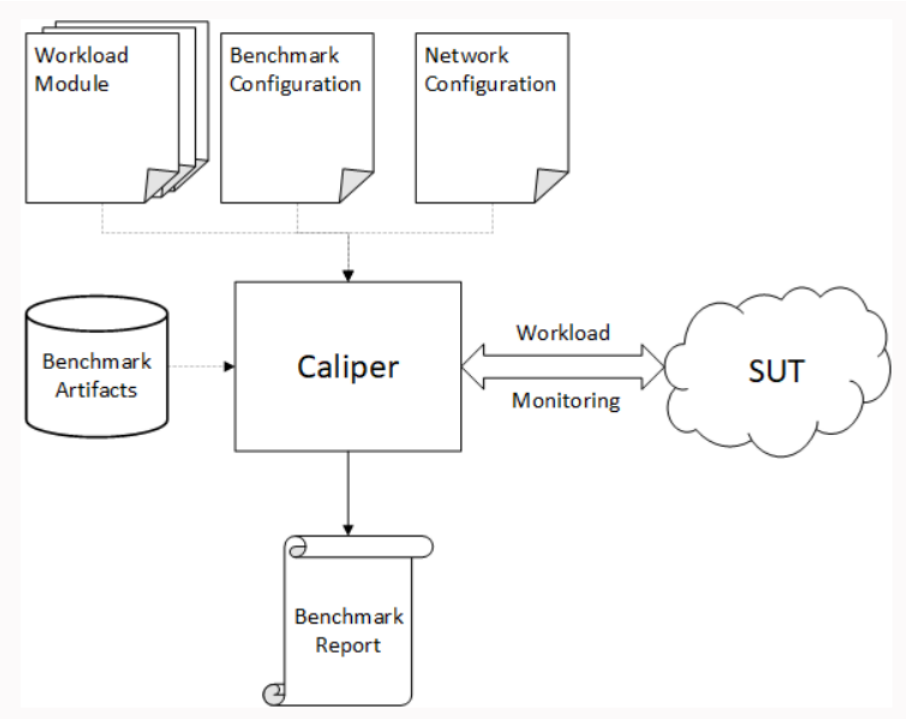


Figura 4.3: Arquitetura da ferramenta *Hyperledger Caliper*, (The Linux Foundation, 2020)

Como é observável na figura 4.3, a *Caliper* necessita de alguns ficheiros, módulos e artefactos para conseguir executar um *benchmark*.

- O **ficheiro de configuração do *benchmark*** descreve a forma como este deve ser executado. É nele que está explicito o número de rondas que a *Caliper* deve efetuar, a taxa de envio de transações e o módulo que gera o conteúdo da transação. Também inclui a monitorização da plataforma em teste. É definido como o "orquestrador do fluxo"do *benchmark*. Na maioria das vezes, este ficheiro é independente do SUT, podendo ser reutilizado para realizar testes a diferentes tipos ou versões deste.

- O **ficheiro de configuração da rede** é específico do sistema a testar. Contém uma descrição da topologia deste, referindo onde se encontram os seus nós, quem está presente na rede e quais os *smart contracts* que a *Caliper* deve implementar ou com os quais tem de interagir.

- Os ***workload modules*** são responsáveis por gerar o conteúdo de uma transação e o enviar. Cada ronda pode ter um módulo diferente associado.

- Os **artefactos de *benchmark*** são aspectos adicionais que podem ser necessários para executar um *benchmark*. São exemplos: materiais necessários para interagir com o SUT, ficheiros de configuração de tempo de execução, entre outros.

Uma das preocupações na construção da *Hyperledger Caliper* foi a escalabilidade do sistema de testes desta ferramenta. A geração de *workloads* numa única máquina pode rapidamente alcançar as limitações de recursos desta. Para que a geração de *workloads* corresponda às características de escalabilidade e desempenho da plataforma a testar, é necessária uma abordagem distribuída.

Assim, esta ferramenta compreende dois processos diferentes:

- O processo mestre (ou ***master process***) - que inicializa o sistema a ser testado e coordena a execução do *benchmark*. É este processo que gera o relatório de desempenho, com base nas estatísticas recolhidas.

- Os processos de trabalho (ou ***worker processes***) - que efetuam a geração de *workloads*, independentemente uns dos outros. Ou seja, mesmo que um destes processos atinja o limite da sua máquina, os outros processos podem continuar. Os *worker processes* são os principais responsáveis pela escalabilidade desta ferramenta.

Na figura 4.4 é ilustrada toda a configuração que se acabou de descrever.

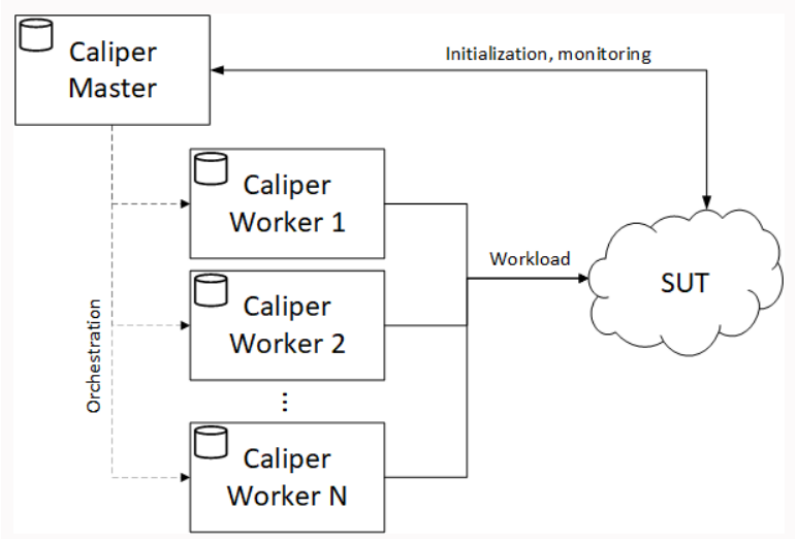


Figura 4.4: Processos da *Hyperledger Caliper*, (The Linux Foundation, 2020)

De seguida são apresentados vários estudos que utilizaram a *Caliper* para proceder à avaliação da performance de plataformas *blockchain*.

Para a realização do trabalho da sua tese de mestrado, (Leppelsack, 2018) utilizou a ferramenta *Hyperledger Caliper* para definir, gerar e executar os *workloads*. O autor baseou-se no trabalho de (Dinh et al., 2017) para a sua elaboração. O estudo foca-se em avaliar a performance de uma *Distributed Ledger Technology (DLT)*, tendo em consideração o *throughput*, a latência das operações de escrita e leitura, o consumo de recursos, o tempo de execução e a taxa de sucesso.

Em (Nasir et al., 2018) foi utilizada uma versão modificada da ferramenta *Hyperledger Caliper* para a realização do seu trabalho. Neste trabalho, foi avaliada a performance da *Hyperledger Fabric*, tendo em consideração a versão 0.6 e a 1.0. Para o estudo destas plataformas, foram recolhidas métricas relativamente à sua taxa de transferência, latência, escalabilidade e tempo de execução. Os autores concluem que a versão 1.0 supera de forma consistente a versão 0.6.

O trabalho realizado por (Baliga et al., 2018) tem como foco a plataforma *Quorum*. De forma a avaliar a sua performance, foi tida em conta a sua taxa de transferência e latência, utilizando a ferramenta *Hyperledger Caliper*. Os autores concluem que o *Raft* e o IBFT são comparáveis no que diz respeito à taxa de transferência de transações. Referem ainda que a utilização de *smart contracts* resulta num *throughput* menor, que se deve às operações de comunicação seguras.

4.2.3 Gauge

A ferramenta *Gauge* consiste num *fork* da *Hyperledger Caliper* (Caliper, 2019), suportando as plataformas *blockchain Hyperledger Fabric*, das versões 1.0 à 1.14, e *Quorum*, na versão 2.0 (Rivankar, 2019).

Os testes que esta permite implementar são divididos em duas categorias:

- ***Controlled Workloads*** - são responsáveis pela recolha de dados relativos à taxa de transferência (*throughput*) e latência na confirmação das transações (Rivankar, 2019);
- ***Micro Benckmarks*** - alterando os parâmetros ao nível das transações e dos *smart contracts*, é conseguido o estudo do impacto destas modificações na latência das transações (Rivankar, 2019).

É ainda possível a realização de testes de *benchmark* relativos à escalabilidade da plataforma, contudo, estes apenas estão disponíveis para a *Hyperledger Fabric* (Rivankar, 2019).

A ferramenta *Gauge* suporta as seguintes métricas, segundo (Rivankar, 2019):

- **Taxa de Transferência** - que corresponde ao número de transações confirmadas com sucesso pela *blockchain*, por segundo (Rivankar, 2019);
- **Latência na Confirmação da Transação** - o tempo que demora desde que um cliente envia uma transação até que recebe a sua confirmação pelo *peer* (Rivankar, 2019);
- **Consumo de Recursos nos *Peers*** - memória e Central Process Unit (CPU) utilizados nos *peers* (Rivankar, 2019).

Na figura 4.5 está representada a arquitetura da ferramenta *Gauge*.

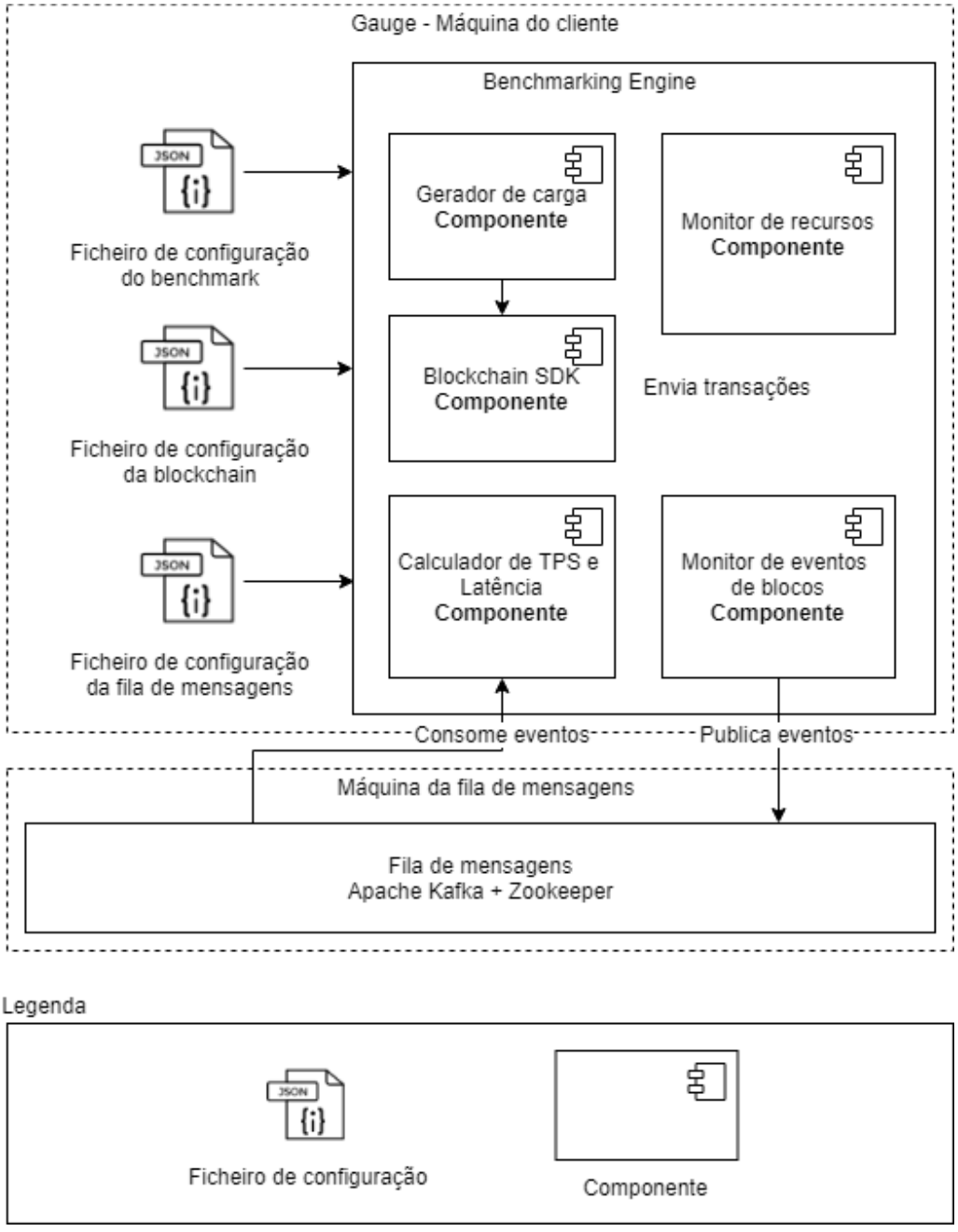


Figura 4.5: Arquitetura da *Gauge*, retirada de (Dias, 2019)

Como é possível observar, a *Gauge* é constituída pelos seguintes componentes:

- **Máquina do Cliente** - armazena todos os arquivos de configuração necessários no ***Benchmarking Engine*** assim como este mecanismo. O *Benchmarking Engine* começa por aceder aos ficheiros de configuração (Solanki, 2018). Posteriormente, o **Gerador de Carga** gera transações para a rede *blockchain* com uma determinada taxa de envio, conforme é especificado nos ficheiros de configuração. Este componente utiliza o ***Blockchain SDK*** para as enviar para a rede *blockchain* (Solanki, 2018). O **Monitor de Recursos**, aquando o envio das transações para a *blockchain*, obtém dados relativos ao consumo de CPU e memória Random Access Memory (RAM) dos nós da rede (Solanki, 2018). De seguida, o **Monitor de Eventos de Blocos** fica à escuta de eventos na *blockchain*. Estes são recebidos quando novos blocos são confirmados, atribuindo-lhes um *timestamp*, que representa a data de confirmação de todas as transações dentro do bloco. Estes blocos "registados"são armazenados na **Fila de Mensagens** até serem lidos pelo **Calculador de TPS e Latência** (Solanki, 2018). Por último, este Calculador lê os blocos que se encontram na fila de mensagens para calcular a taxa de transferência e a latência das transações (Solanki, 2018).
- **Máquina da Fila de Mensagens** - é implementada utilizando o *Apache Kafka + ZooKeeper*. Esta pode ser executada numa ou mais máquinas, de forma a melhorar a eficiência da fila de mensagens. Como mencionado acima, esta armazena os blocos "registados"até que estes sejam lidos pelo Calculador de TPS e Latência (Solanki, 2018).

4.3 Análise Geral

Nesta secção é feita uma análise geral das ferramentas de testes *benchmark*.

Na tabela 4.1 encontram-se representadas as métricas que cada uma das ferramentas permite recolher. Estão estão assinaladas com um ✓no caso de ser possível a sua recolha ou com um ✗ caso contrário.

Métricas	**Ferramenta**		
	BLOCKBENCH	Hyperledger Caliper	Gauge
Taxa de Transferência	✓	✓	✓
Latência	✓	✓	✓
Consumo de Recursos	✗	✓	✓
Escalabilidade	✓	✓	✓
Taxa de Sucesso	✗	✓	✓
Tolerância a Falhas	✓	✗	✗
Tempo de Execução	✗	✓	✓
Segurança	✓	✗	✗

Tabela 4.1: Análise das ferramentas relativamente às métricas que recolhe

A tabela 4.2 apresenta as caraterísticas de cada um dos projetos. O facto de ser Open-Source é um dos aspectos mais fulcrais, pois assim é possível ter acesso ao seu código fonte, sendo permitida a sua alteração conforme as necessidades. Observando a tabela, verifica-se que todas elas fazem uso da licença *Apache* 2.0. É também importante que o código fonte esteja em constante atualização, pois desta forma percebe-se que a ferramenta em questão continua a ser utilizada. O facto das ferramentas apresentarem uma documentação bem estruturada e a existência de comentários no código fonte, são fatores que facilitam a compreensão do funcionamento da ferramenta.

Ferramenta	***OpenSource***	**Atualização do projeto**	**Compreensão**
BLOCKBENCH	*Apache License* 2.0	Repositório atualizado em Março 2020	Documentação escassa; poucos comentários no código
Hyperledger Caliper	*Apache License* 2.0	Repositório atualizado mensalmente	Documentação bem organizada e explicada; o código apresenta alguns comentários
Gauge	Descontinuada (*Apache License* 2.0)	Descontinuada	Descontinuada (Documentação semelhante à da *Hyperledger Caliper*)

Tabela 4.2: Análise relativamente às caraterísticas das ferramentas

Observando as tabelas é possível notar que as ferramentas *Hyperledger Caliper* e *Gauge* são em tudo semelhantes, o que provém do facto de a segunda ser um *fork* da primeira. As principais diferenças estão nas plataformas *blockchain* que cada uma delas suporta e na sua atualização. A *Gauge* é uma ferramenta que se encontra descontinuada, não sendo possível o acesso ao seu repositório, visto que este já não está disponível.

4.4 Conclusão

Neste capítulo foi realizado o estudo de diversos trabalhos existentes na área de *benchmarking* de plataformas *blockchain* bem como apresentadas diversas ferramentas que permitem a avaliação das mesmas.

Foi tomada a decisão de utilizar a ferramenta *Hyperledger Caliper* por se tratar de um projeto *OpenSource* e em constante evolução. O suporte que esta tem a diversas plataformas *blockchain*, a documentação disponível para a sua compreensão e a disponibilidade para ajudar e esclarecer qualquer tipo de dúvidas apresentada por partes dos seus contribuidores, foram também outros fatores que influenciaram esta escolha.

Esta página foi propositadamente deixada em branco.

Capítulo 5

Ambiente de Teste

Neste capítulo procede-se a uma explicação da forma como foi preparada toda a execução dos testes de *benchmark* a realizar às plataformas *blockchain* escolhidas. Em primeiro lugar, na secção 5.1, é descrito o ambiente de testes implementado, as suas caraterísticas e a sua organização. Posteriormente, na secção 5.2 são definidos os testes que se pretende realizar, de forma a atingir os objetivos propostos neste documento.

5.1 Preparação do Ambiente de Testes

Para a realização dos testes de *benchmark* foi necessário criar um ambiente de execução que suportasse todo o sistema. Este encontra-se dividido em três máquinas virtuais, presentes na *Cloud* do Departamento de Engenharia Informática (DEI). Cada uma destas possui 300 GB de memória em disco, 16 GB de memória RAM e 8 vCPU's, correndo o sistema operativo Ubuntu 16.04. Em cada uma destas máquinas, foi instalada a ferramenta *Hyperledger Caliper* e a plataforma *blockchain* que se pretendia estudar.

Após a instalação da *Caliper* e da conexão desta com a respetiva plataforma *blockchain* em cada uma das máquinas (o que não foi possível em relação à *Quorum*), procedeu-se ao acesso ao repositório que contém *benchmarks* que podem ser usados pela ferramenta em questão. A conexão da ferramenta com as plataformas é feita fazendo a ligação do *Caliper CLI* com o SDK da plataforma (Caliper, 2019).

Ao utilizar esta ferramenta, para a execução de uma ronda de teste, é gerado e guardado um ficheiro no formato "*.log*"onde são apresentadas as seguintes métricas e informações que a *Hyperledger Caliper* permite recolher:

- *Succ* - Número inteiro, simboliza o número total de transações submetidas com sucesso;
- *Fail* - Número inteiro, representa o número total de transações que não foram submetidas com sucesso;
- Taxa de Sucesso - Representa a percentagem de transações que foram submetidas com sucesso;
- *Throughput* - Taxa de transferência, apresentada em transações por segundo (TPS);
- Latência Média - Medida em segundos (s), tempo que ocorre desde o envio de uma transação até ao tempo de confirmação da mesma;

- Percentagem de CPU utilizada (em média).

Para a elaboração dos testes foi tida em consideração a possibilidade de existirem *outliers* que pudessem influenciar os resultados obtidos. De forma a ser possível minimizar ou eliminar o aparecimento destes dados, antes de ser executado um determinado teste, a plataforma é submetida a um período de "aquecimento". Este período consiste no envio de 5000 transações para a plataforma, a um ritmo de 50 Transações Por Segundo (TPS) no caso da plataforma *Hyperledger Fabric* e 500 TPS na plataforma FISCO BCOS. Foram definidos estes ritmos para a fase de aquecimento por se tratarem dos ritmos de envio mais baixos utilizados na execução dos testes. Por sua vez, estes *send rates* foram escolhidos após a execução de alguns testes exemplo, realizados anteriormente à execução dos testes, de forma a verificar qual o comportamento que cada uma das plataformas apresentava. Observando este comportamento, realizando apenas testes experimentais na ordem de perceber como se iniciava a plataforma *blockchain* e quais os testes que era permitido executar por parte da ferramenta *Hyperledger Caliper*, foi possível retirar algumas conclusões relativamente a cada uma das plataformas. No caso da *Hyperledger Fabric*, foi possível notar que, aumentando o ritmo de envio para a *blockchain*, o *throughput* alcançado por esta não excedia as 500 TPS. O mesmo se verificou para a FISCO BCOS, nas 5000 TPS. Desta forma, os testes foram definidos tendo em conta estes aspectos, considerando desnecessário a utilização de *send rates* superiores a estes valores aquando da realização dos testes.

Os resultados provenientes das rondas de aquecimento, não são considerados nos resultados finais. Este período permite que a plataforma atinja um estado estável no qual é menos provável o aparecimento de desvios nos resultados. Outra abordagem utilizada para eliminar *outliers* foi a repetição de cada teste dez vezes.

Para a elaboração dos testes relativos à plataforma *blockchain* FISCO BCOS foram utilizadas as configurações da rede que se encontravam disponíveis para a execução dos testes de *benchmark* no repositório da *Hyperledger Caliper* - (Klenik et al., 2020). Esta rede é constituída por 4 nós e utiliza *LevelDB* para a persistência de dados. Com a utilização da ferramenta *Hyperledger Caliper* apenas foi possível verificar o funcionamento desta plataforma com estas configurações e exercendo sobre ela operações básicas de leitura e escrita na *blockchain*.

Relativamente à plataforma *Hyperledger Fabric*, a ferramenta já se encontra preparada para modificar alguns aspetos da rede. Nos testes mais gerais, de forma a conseguir uma comparação com a FISCO BCOS, optou-se por utilizar uma rede com duas organizações, cada uma delas com um *peer*, sem mecanismo de consenso, utilizando *LevelDB* para a persistência de dados.

Posteriormente, definiu-se que seria testada a diferença de utilizar ou não o mecanismo de consenso *Raft* e a alteração da base de dados utilizada, ou seja, a diferença entre a escolha de *LevelDB* ou *CouchDB*. A ferramenta *Hyperledger Caliper* também fornece módulos semelhantes aos disponibilizados para a FISCO BCOS, ou seja, permite efetuar operações básicas de leitura e escrita sobre a *blockchain*, também no caso da *Fabric*.

De forma a automatizar todo o processo de *benchmark*, foram criados diferentes *scripts*:

- run.sh - que corresponde à realização dos testes mais básicos efetuados sobre as plataformas FISCO BCOS e *Hyperledger Fabric*;
- run_raft.sh - que permite a execução dos testes à *Hyperledger Fabric* utilizando o mecanismo de consenso *Raft*;

- run_couch.sh - que diz respeito à alteração da base de dados *LevelDB* para a *CouchDB*, na execução dos testes sobre a *Hyperledger Fabric*.

Nestes *scripts* está contida a informação do ficheiro de configuração do *benchmark* - onde é definido o tipo de operação que se pretende realizar sobre a *blockchain* bem como definido o número de transações que se pretende enviar - e do ficheiro de configuração da rede - onde se encontra descrito o número de organizações e *peers* que estas contêm, o mecanismo de consenso utilizado e a base de dados usada para a persistência dos dados.

5.2 Definição dos Testes a realizar

Nesta secção são definidos, descritos e apresentados os testes de *benchmark* que se pretende executar. A cada teste corresponde uma tabela, que se encontra dividida em cinco partes. A primeira corresponde ao Nome do Teste. De seguida, é atribuído um identificador a este. O objetivo principal do teste, bem como os parâmetros de *input* e *output* que são utilizados e obtidos, são também referidos nas tabelas.

Os testes vão incidir nas métricas mencionadas anteriormente pois tratam-se de valores que são possíveis recolher em ambas as plataformas *blockchain* estudadas. A escolha destes testes passou pela análise daqueles que se encontram disponíveis na ferramenta de *benchmark*, ou seja, no seu repositório - (Klenik et al., 2020) - bem como pelas conclusões retiradas aquando da realização dos testes experimentais que foram feitos para entender o funcionamento da ferramenta e de cada uma das plataformas *blockchain*. Como foi mencionado anteriormente, foi com a ajuda destes testes que foram definidos os *send rates* a utilizar para cada uma das plataformas.

Nome do Teste	Teste de performance realizando operações de leitura na *blockchain*
ID	T1
Descrição	Este teste consiste na análise da performance das plataformas *Hyperledger Fabric* e FISCO BCOS quando estas são submetidas a operações de leitura na *blockchain*.
Parâmetros de *Input*	*Send Rate* - que varia entre as 50 e as 500 TPS (incremento de 50 TPS), no caso da *Hyperledger Fabric* e entre as 500 e 5000 TPS (incremento de 500 TPS), no caso da FISCO BCOS; Número total de transações enviadas para a *blockchain* - varia entre 2500 e 30000 transações (incremento de 2500 transações), para ambas as plataformas.
Parâmetros de *Output*	Latência média; *Throughput*.

Tabela 5.1: Definição do Teste T1

Nome do Teste	Teste de consumo de recurso realizando operações de leitura na *blockchain*
ID	T2
Descrição	Este teste consiste na análise do consumo de recursos das plataformas *Hyperledger Fabric* e FISCO BCOS quando estas são submetidas a operações de leitura na *blockchain*.
Parâmetros de *Input*	*Send Rate* - que varia entre as 50 e as 500 TPS (incremento de 50 TPS), no caso da *Hyperledger Fabric* e entre as 500 e 5000 TPS (incremento de 500 TPS), no caso da FISCO BCOS; Número total de transações enviadas para a *blockchain* - varia entre 2500 e 30000 transações (incremento de 2500 transações), para ambas as plataformas.
Parâmetros de *Output*	Percentagem de CPU utilizada

Tabela 5.2: Definição do Teste T2

Nome do Teste	Teste de performance realizando operações de escrita na *blockchain*
ID	T3
Descrição	Este teste consiste na análise da performance das plataformas *Hyperledger Fabric* e FISCO BCOS quando estas são submetidas a operações de escrita na *blockchain*.
Parâmetros de *Input*	*Send Rate* - que varia entre as 50 e as 500 TPS (incremento de 50 TPS), no caso da *Hyperledger Fabric* e entre as 500 e 5000 TPS (incremento de 500 TPS), no caso da FISCO BCOS; Número total de transações enviadas para a *blockchain* - varia entre 2500 e 30000 transações (incremento de 2500 transações), para ambas as plataformas.
Parâmetros de *Output*	Latência média; *Throughput*.

Tabela 5.3: Definição do Teste T3

Nome do Teste	Teste de consumo de recursos realizando operações de escrita na *blockchain*
ID	T4
Descrição	Este teste consiste na análise do consumo de recursos das plataformas *Hyperledger Fabric* e FISCO BCOS quando estas são submetidas a operações de escrita na *blockchain*.
Parâmetros de *Input*	*Send Rate* - que varia entre as 50 e as 500 TPS (incremento de 50 TPS), no caso da *Hyperledger Fabric* e entre as 500 e 5000 TPS (incremento de 500 TPS), no caso da FISCO BCOS; Número total de transações enviadas para a *blockchain* - varia entre 2500 e 30000 transações (incremento de 2500 transações), para ambas as plataformas.
Parâmetros de *Output*	Percentagem de CPU utilizada

Tabela 5.4: Definição do Teste T4

Nome do Teste	Teste à utilização do mecanismo de consenso *Raft*
ID	T5
Descrição	Este teste diz respeito à análise do comportamento da plataforma *Hyperledger Fabric* quando é ou não utilizado o mecanismo de consenso *Raft*.
Parâmetros de *Input*	*Send Rate* - que varia entre as 50 e as 500 TPS (incremento de 50 TPS); Número total de transações enviadas para a *blockchain* - varia entre 2500 e 30000 transações; Mecanismo de Consenso - Nenhum ou *Raft*.
Parâmetros de *Output*	Latência média; *Throughput*; Percentagem de CPU utilizada; Número total de transações bem sucedidas.

Tabela 5.5: Definição do Teste T5

Nome do Teste	Teste à utilização de diferentes bases de dados
ID	T6
Descrição	Com a realização deste teste é pretendido estudar o comportamento da plataforma *Hyperledger Fabric* quando esta utiliza *LevelDB* ou *CouchDB* para a persistência de dados.
Parâmetros de *Input*	*Send Rate* - que varia entre as 50 e as 500 TPS (incremento de 50 TPS); Número total de transações enviadas para a *blockchain* - varia entre 2500 e 30000 transações (incremento de 2500 transações); Base de Dados - *LevelDB* ou *CouchDB*.
Parâmetros de *Output*	Latência média; *Throughput*; Percentagem de CPU utilizada; Número total de transações bem sucedidas.

Tabela 5.6: Definição do Teste T6

5.3 Resumo do Capítulo

Neste capítulo foi apresentado o ambiente de testes que foi utilizado para a realização dos testes de *benchmark* efetuados para estudar as plataformas *blockchain Hyperledger Fabric* e FISCO BCOS.

Foram também definidos os testes a efetuar, referindo os parâmetros de *input* para a sua execução e os de *output* esperados. No total, foram definidos seis testes. Os quatro primeiros referem-se à análise e comparação do desempenho das plataformas *Hyperledger Fabric* e FISCO BCOS. O quinto teste diz respeito à utilização ou não do mecanismo de consenso *Raft*, na *Fabric*. O sexto e último permite analisar as diferenças no desempenho da plataforma *Hyperledger Fabric* quando esta utiliza *LevelDB* ou *CouchDB* para a persistência de dados.

Esta página foi propositadamente deixada em branco.

Capítulo 6

Benchmarking de Plataformas *Blockchain*

Neste capítulo são apresentados os resultados obtidos após a execução dos testes de *benchmark*. Em primeiro lugar na secção 6.1, é apresentada a forma como estes foram organizados. As secções 6.2 a 6.7 dizem respeito a cada teste individual e seus resultados, sempre complementados com uma análise e discussão dos mesmos. Por fim, na secção 6.8, é feito um pequeno resumo que contém a conclusão geral dos *benchmarks* efetuados, comparando os resultados obtidos com outros que foram alcançados por outros autores.

6.1 Organização dos Testes

Todos os seis testes propostos para avaliar e comparar as plataformas *blockchain* foram efetuados. Cada teste foi repetido dez vezes, como tinha sido proposto, de forma a eliminar possíveis dados incorrectos originados por perturbações na rede.

Para analisar o comportamento da plataforma FISCO BCOS, foram definidos diferentes *Send Rates*, que variam entre as 500 e as 5000 TPS, incrementando este valor de 500 em 500 TPS. Para cada *Send Rate*, foi também variado o número total de transações enviadas para a *blockchain*, começando pelas 2500 até às 30000 transações, com um incremento de 2500 transações. Foram definidos estes valores de *Send Rate* por se verificar que esta plataforma não atinge valores superiores a 4000 TPS.

No que diz respeito à plataforma *Hyperledger Fabric*, foi utilizado o mesmo método para os testes, com a diferença dos valores de *Send Rate*. Para esta plataforma, iniciou-se com 50 TPS até 500 TPS, incrementando este valor de 50 em 50 TPS. Estes valores foram escolhidos pelo mesmo motivo mencionado anteriormente, ou seja, por se verificar que esta plataforma não atinge valores de *Send Rate* superiores a 300 TPS.

Relativamente à exposição dos resultados obtidos, serão apresentados gráficos dois a dois. Cada um destes representa uma plataforma *blockchain* diferente, *Hyperledger Fabric* e FISCO BCOS, no caso das secções 6.2, 6.3, 6.4 e 6.5. Na secção 6.6, uma das representações corresponde ao comportamento da plataforma quando não é utilizado o mecanismo de consenso *Raft* e o outro quando este é utilizado, na plataforma *Fabric*. Na secção 6.7, um dos gráficos corresponde à utilização da *LevelDB* e o outro ao uso de *CouchDB*, na plataforma *Fabric*.

De forma a facilitar a leitura e interpretação das ilustrações, após a apresentação da-

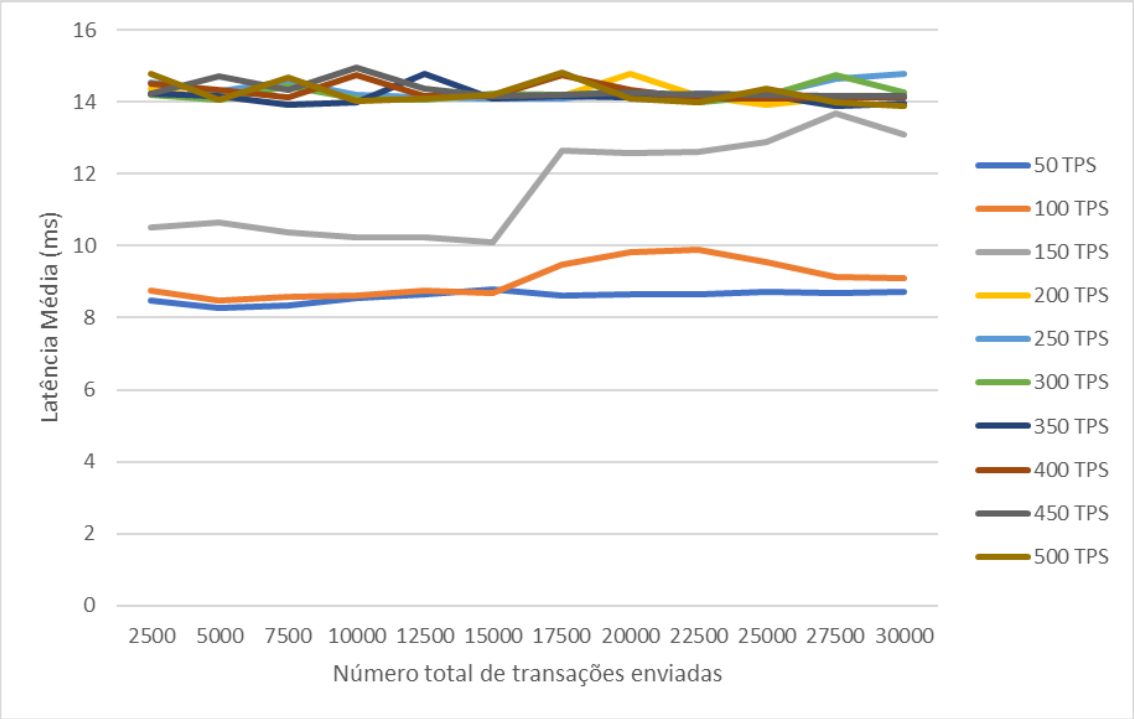
queles onde estão representados todos os resultados obtidos, são exibidos outros, focados apenas numa parte do gráfico inicial, onde se encontra representado o intervalo de confiança, com o valor de alfa igual a 0,05.

Nas secções que se seguem são apresentados os resultados dos testes realizados e feita a análise dos mesmos.

6.2 T1 - Teste de performance realizando operações de leitura na *blockchain*

O teste T1 tem como principal objetivo estudar e comparar o desempenho das plataformas *Hyperledger Fabric* e FISCO BCOS quando sobre elas são exercidas operações de leitura na *blockchain*. Com este propósito, foi utilizada uma função simples que apenas lê os dados guardados na *blockchain*.

Na figura 6.1 encontra-se representado o comportamento da latência média em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* e o segundo à FISCO BCOS.



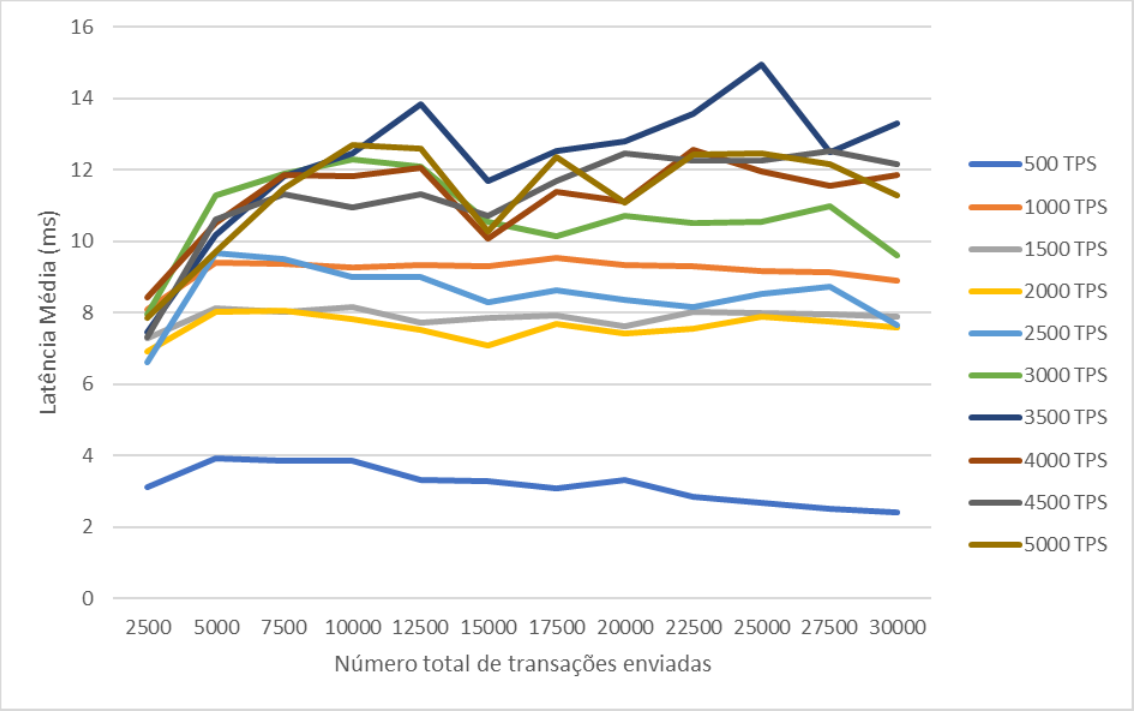


Figura 6.1: T1 - Comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric, Fundo: FISCO BCOS

A partir dos gráficos, é possível observar que, em ambas as plataformas, os valores de latência média não ultrapassam os 16 milissegundos. Em relação à *Fabric*, quando submetida a *Send Rates* entre as 50 e as 150 TPS, esta demora menos tempo a responder aos pedidos de transações, apresentando assim valores de latência média mais baixos. Contudo, é de salientar o comportamento desta plataforma quando atinge as 15000 transações enviadas, com um *send rate* de 150 TPS. A partir deste valor, a plataforma possivelmente entra em sobrecarga e aumenta, de forma significativa, o tempo de resposta aos pedidos que lhe são efetuados. A partir das 150 TPS, esta plataforma estabiliza, sendo que, mesmo alterando o *send rate* e o número de transações que são enviadas para a *blockchain*, a sua latência média mantém-se mais ou menos constante, entre os 14 e os 15 milissegundos. No que diz respeito à FISCO BCOS, é notório que quando esta é submetida a um *send rate* de 500 TPS, esta plataforma responde rapidamente aos pedidos de transações que lhe são feitos. Entre as 1000 TPS e as 2500 TPS apresenta valores de latência média entre os 7 e os 10 milissegundos. Para *send rates* mais elevados, a latência média também aumenta.

De seguida, na figura 6.2, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

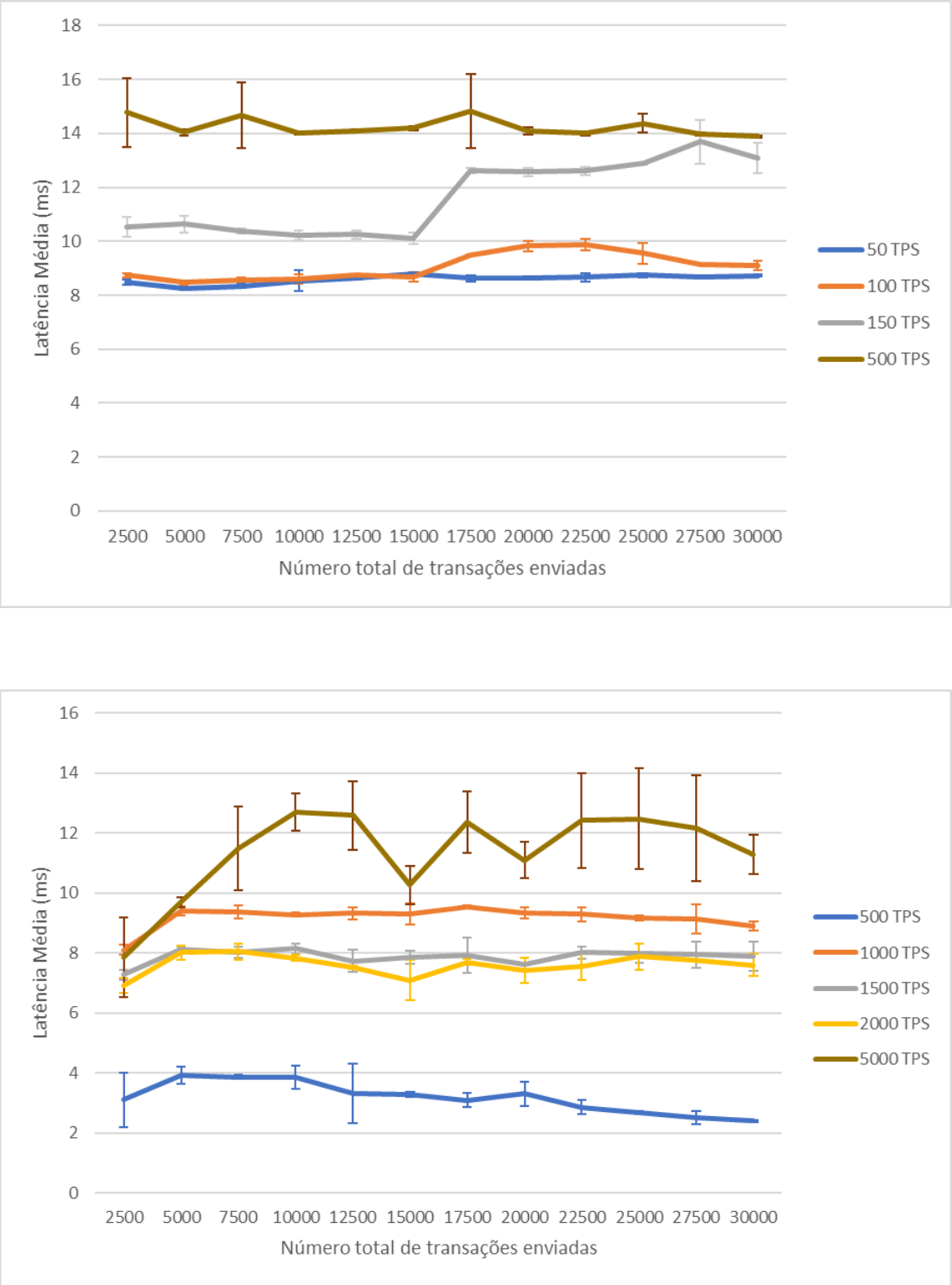


Figura 6.2: T1 - Representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric, Fundo: FISCO BCOS

Com a representação dos intervalos de confiança, é possível verificar que estes são menores na plataforma *Hyperledger Fabric*, com a excepção dos resultados obtidos para um *send rate* de 500 TPS. A FISCO BCOS apresenta valores de intervalo de confiança maiores, principalmente para um *send rate* de 5000 TPS.

Na figura 6.3 é apresentado o comportamento do *throughput* em função do *send rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Hyperledger Fabric* e o segundo à FISCO BCOS. É preciso ter em consideração que os valores apresentados no eixo vertical (yy) são distintos.

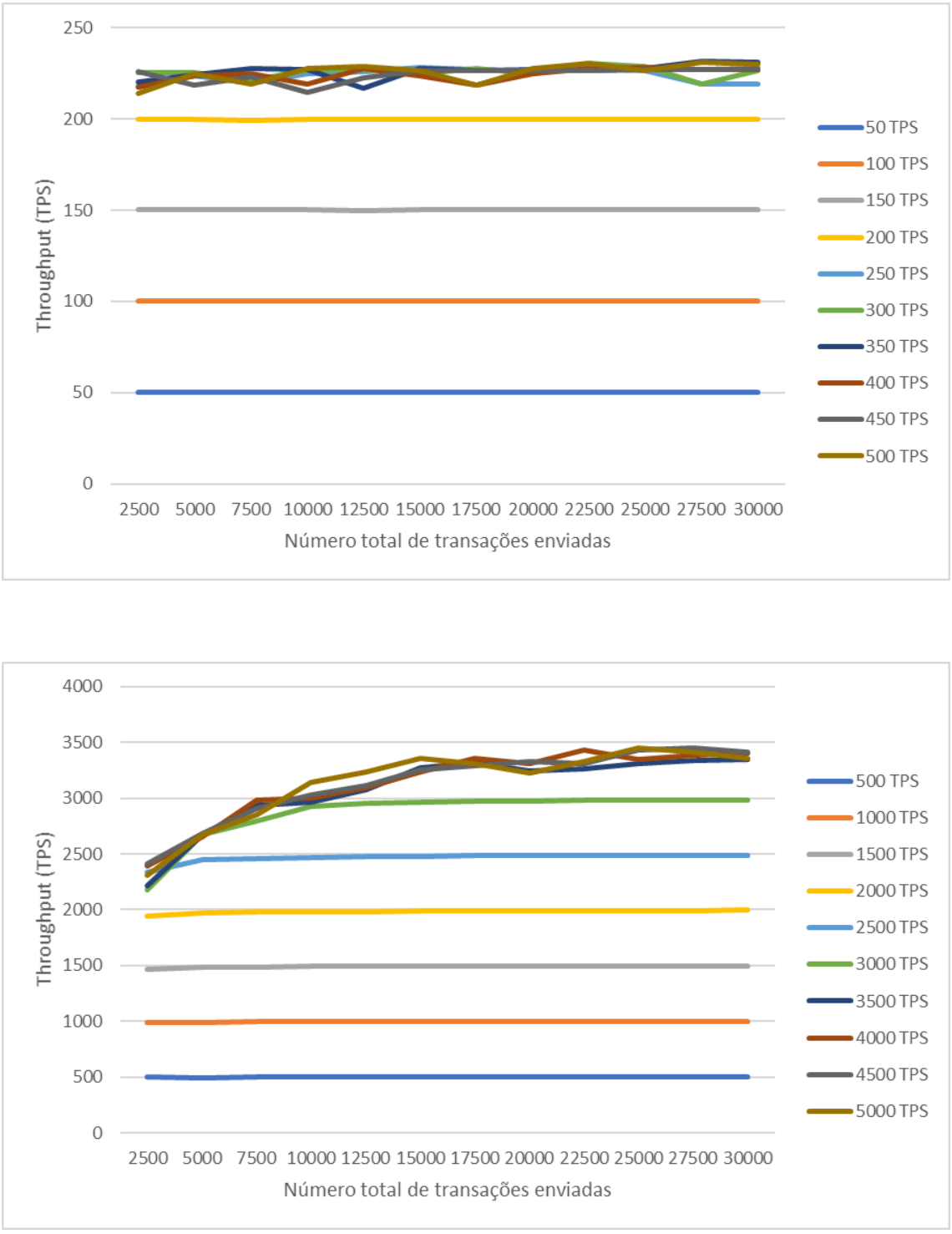


Figura 6.3: T1 - Comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric, Fundo: FISCO BCOS

Analisando os gráficos verifica-se que a plataforma FISCO BCOS atinge valores de *throughput* muito superiores aos alcançados pela *Hyperledger Fabric*. Em ambas as plataformas é notório um momento em que o *throughput* não consegue acompanhar o *send rate*. No caso da *Fabric*, este surge a partir das 200 TPS. No caso da FISCO BCOS, a partir das 2500 TPS. Nas 3000 TPS verifica-se que ainda é possível este acompanhamento a partir das 10000 transações enviadas para a *blockchain*. A plataforma *Fabric* atinge um *throughput* constante por volta das 220 TPS, a partir do *send rate* de 200 TPS. Para *send rates* superiores a 3000 TPS, a FISCO BCOS apresenta valores de *throughput* alcançados crescentes, até às 15000 transações enviadas. Posteriormente, apresenta um valor de *throughput* mais ou menos constante, por volta das 3500 TPS, a partir do *send rate* de 3000 TPS.

De seguida, na figura 6.4, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

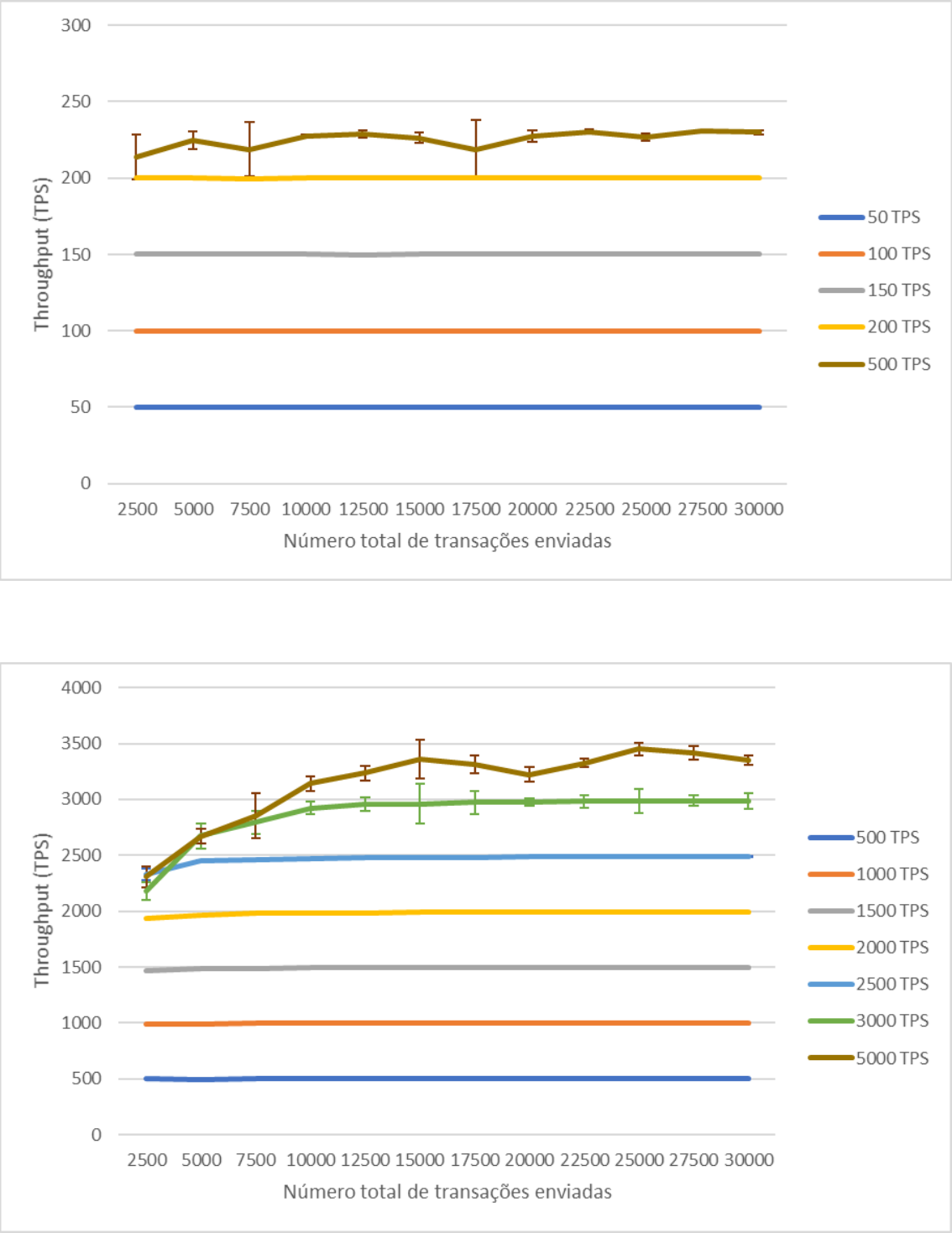


Figura 6.4: T1 - Representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric, Fundo: FISCO BCOS

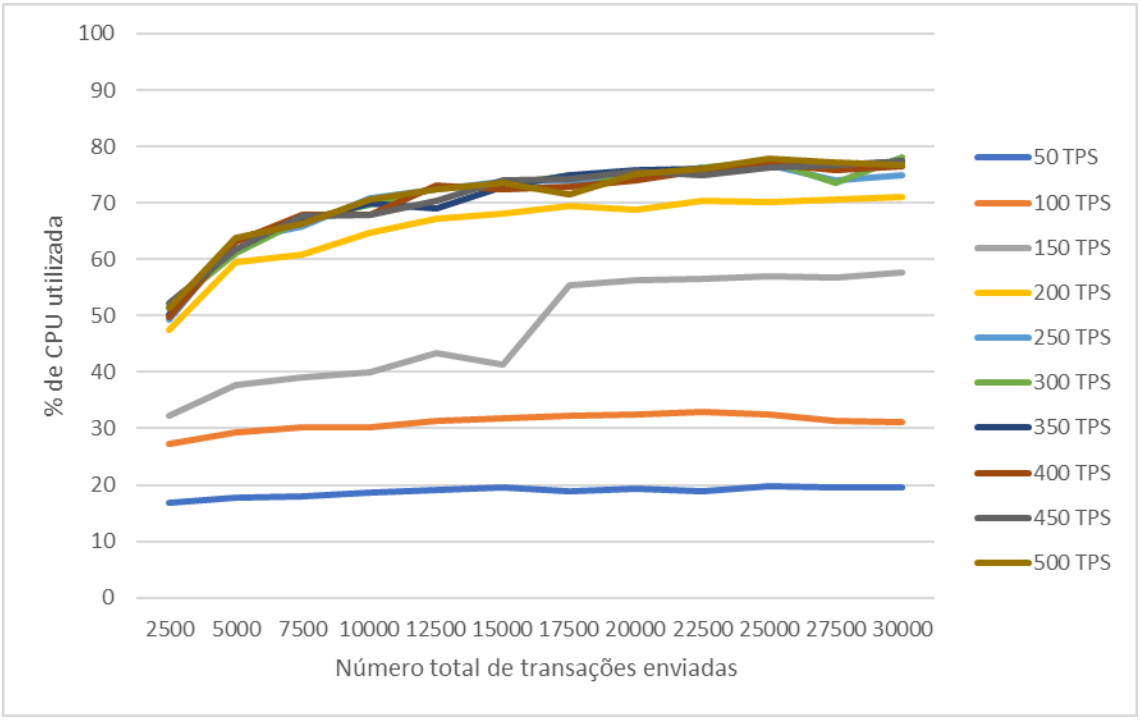
Com a representação dos intervalos de confiança é possível observar que o *send rate* de 500 TPS e 5000 TPS, na *Fabric* e na FISCO BCOS, respetivamente, são os valores que apresentam intervalos de confiança maiores.

Em suma, com este testes foi possível verificar que a plataforma FISCO BCOS apresenta valores de *Throughput* muito superiores aos obtidos na *Fabric*. A plataforma *Fabric* apresenta um comportamento muito estável, enquanto que a FISCO BCOS demonstra algumas variações. Em relação ao valores de latência média obtidos, a *Hyperledger Fabric* apresenta um comportamento mais uniforme que a FISCO BCOS.

6.3 T2 - Teste de consumo de recursos realizando operações de leitura na *blockchain*

O teste T2 tem como principal objetivo estudar e comparar o consumo de recursos das plataformas *Hyperledger Fabric* e FISCO BCOS quando sobre elas são exercidas operações de leitura na *blockchain*. Com este propósito, foi utilizada uma função simples que apenas lê os dados guardados na *blockchain*.

Na figura 6.5 encontra-se representada a percentagem de CPU utilizada em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* e o segundo à FISCO BCOS.



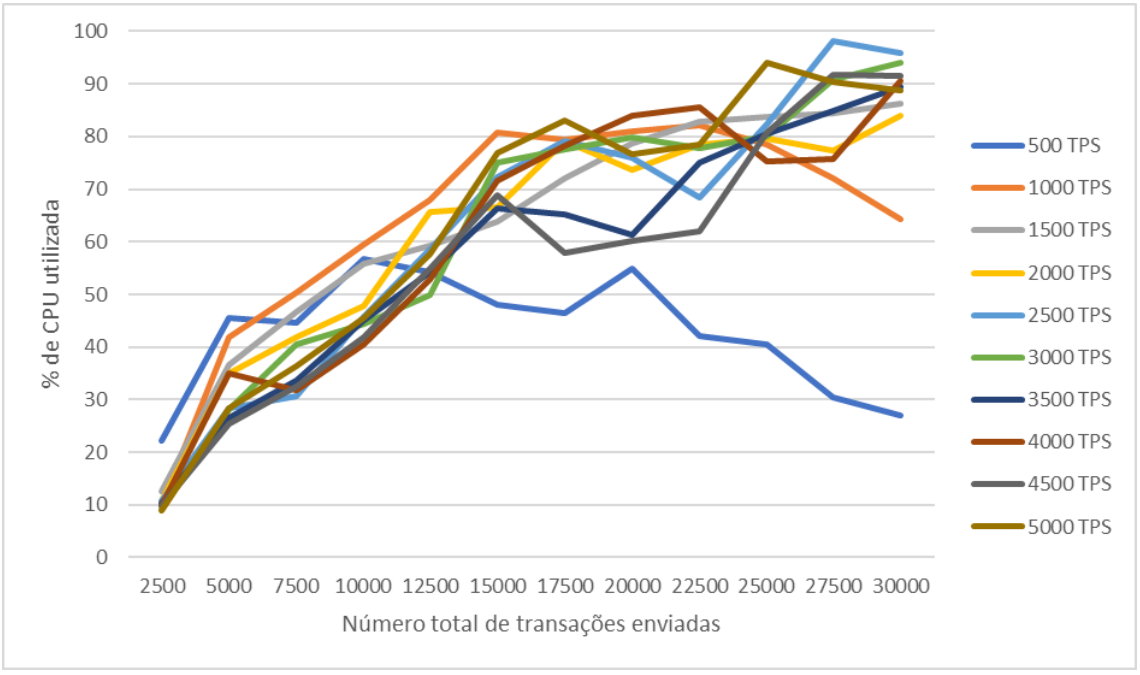
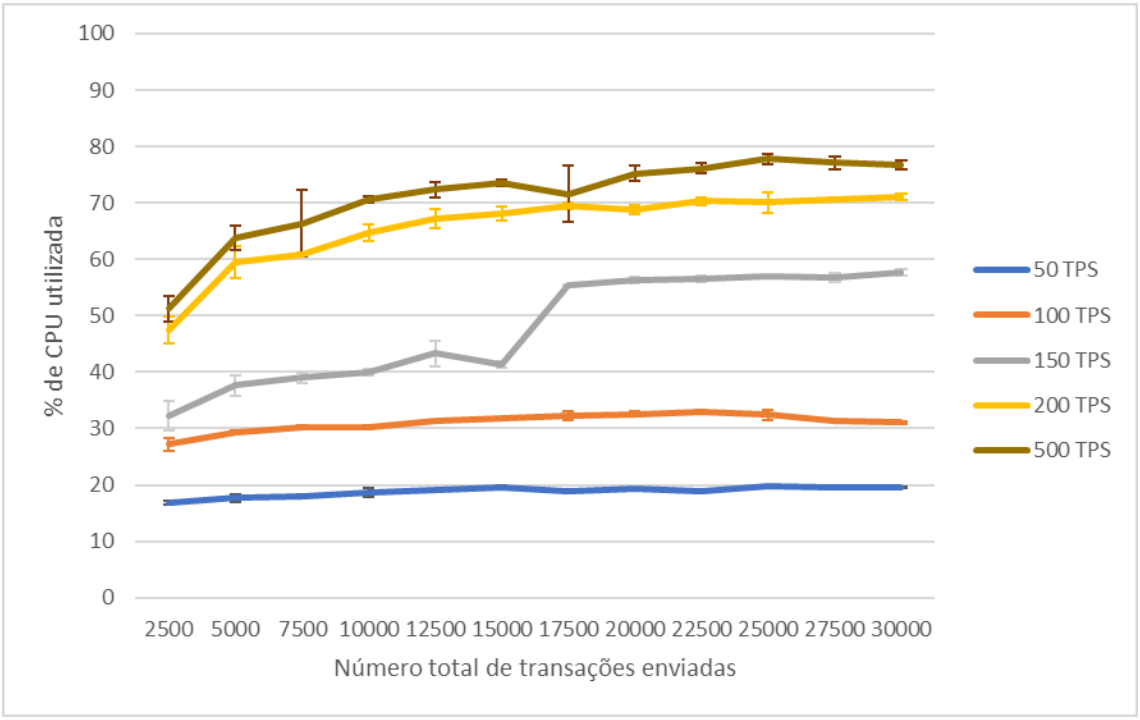


Figura 6.5: T2 - Percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric, Fundo: FISCO BCOS

Tendo em consideração os gráficos apresentados, verifica-se que a plataforma *Fabric* consome uma percentagem de recursos computacionais menor quanto menor for o *send rate* imposto. Por outro lado, a FISCO BCOS apresenta um comportamento mais ou menos semelhante independentemente do *send rate*. Analisando os resultados, observa-se que a percentagem de CPU utilizada pela FISCO BCOS é superior à utilizada pela *Fabric*, atingindo valores perto dos 100%, enquanto que a *Fabric* apenas chega perto dos 80%. É de observar, na FISCO BCOS que, quando o *send rate* é de 500 TPS, a percentagem de CPU utilizada diminui, comportamento que é bastante distinto dos restantes *send rates*.

De seguida, na figura 6.6, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.



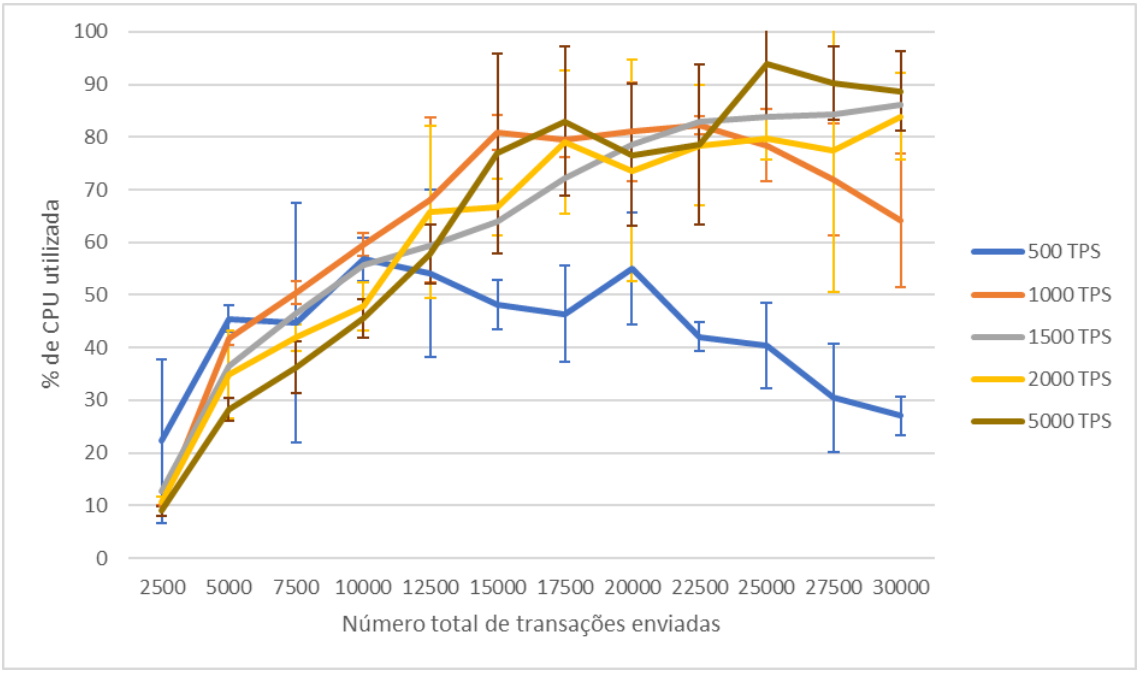


Figura 6.6: T2 - Representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric, Fundo: FISCO BCOS

Observando os gráficos acima é possível verificar que os intervalos de confiança para os valores obtidos da *Fabric* são pequenos. No entanto, no caso da FISCO BCOS, os intervalos são grandes, o que indica uma grande variação nos valores recolhidos.

6.4 T3 - Teste de performance realizando operações de escrita na *blockchain*

O teste T3 tem como principal objetivo estudar e comparar a performance das plataformas *Hyperledger Fabric* e FISCO BCOS quando sobre elas são realizadas operações de escrita na *blockchain*. Com este propósito, , no caso da *Fabric*, foi utilizada uma função que simula a transferência de uma certa quantia de dinheiro de uma conta para outra. No caso da FISCO BCOS, recorreu-se a uma função que altera a quantia de dinheiro de uma conta. Desta forma, é possível analisar o comportamento de cada uma das plataformas, no entanto, não é possível fazer uma comparação direta entre as mesmas, por se tratarem de operações de escrita distintas.

Na figura 6.7 encontra-se representado o comportamento da latência média em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* e o segundo à FISCO BCOS.

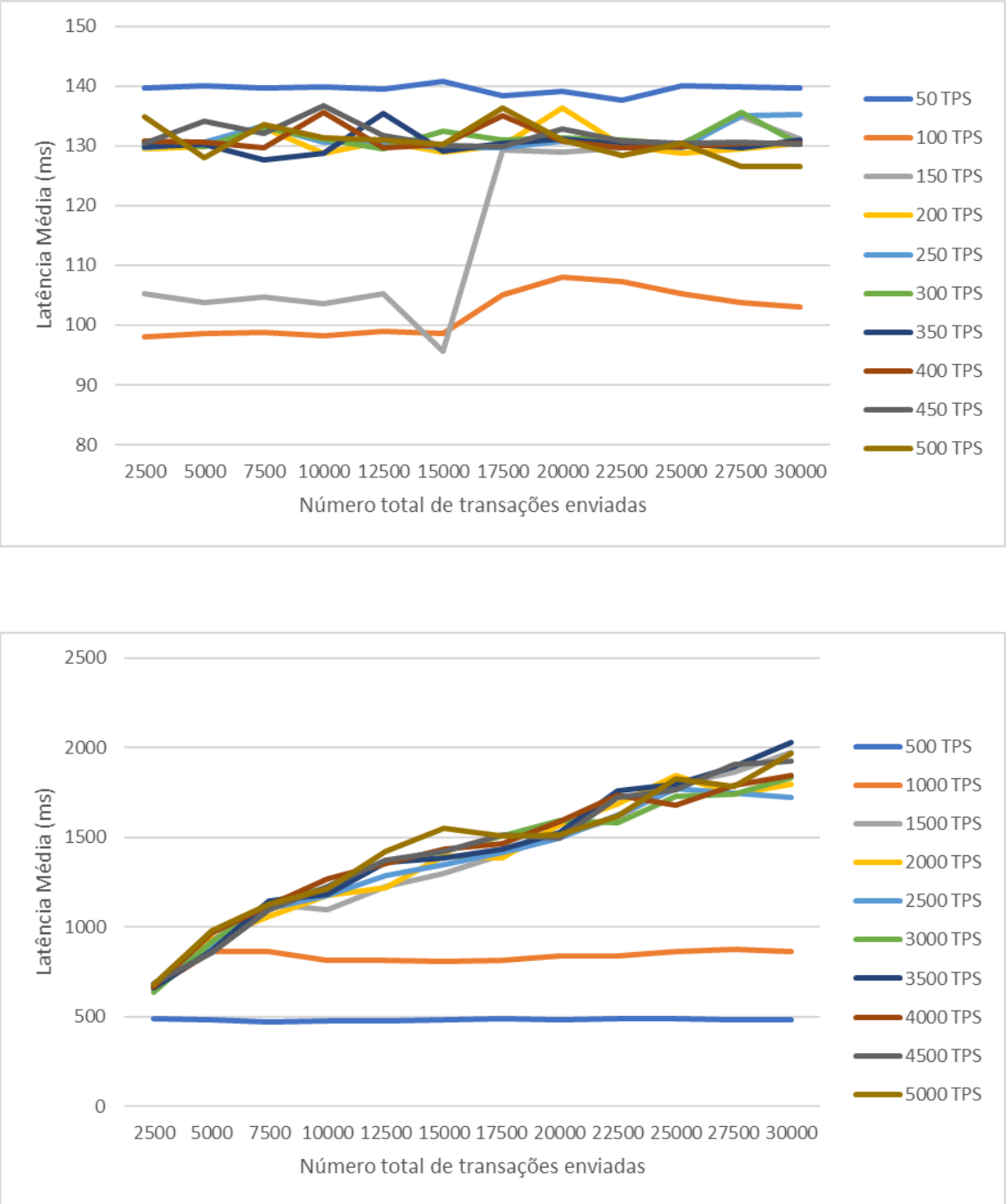


Figura 6.7: T3 - Comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric, Fundo: FISCO BCOS

Observando os gráficos acima, e tendo em consideração os valores distintos dos eixos dos mesmos, é notório que a plataforma FISCO BCOS apresenta valores de latência média muito superiores aos obtidos da *Fabric*. Os valores de latência média, na *Fabric* tornam-se mais ou menos constantes entre os 130 e 140 milissegundos, com *send rate* superior a 150 TPS. É de verificar que, com um *send rate* de 50 TPS, a latência média é superior à de qualquer outro *send rate*. Isto pode ser provocado pelo facto de, sendo o *send rate* baixo, a plataforma não necessitar de tratar os pedidos tão rápido, pois chegam poucos de cada vez. A subida abrupta dos valores de latência no *send rate* de 150 TPS pode dever-se a uma

sobrecarga na rede, a partir das 15000 transações enviadas, em que o sistema não conseguiu lidar com todas. Na FISCO BCOS, os valores médios de latência vão aumentando, para *send rates* superiores a 1000 TPS, apresentando valores semelhantes. Nesta plataforma, quando o *send rate* é de 500 e 1000 TPS, os pedidos são respondidos mais rápido.

De seguida, na figura 6.8, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

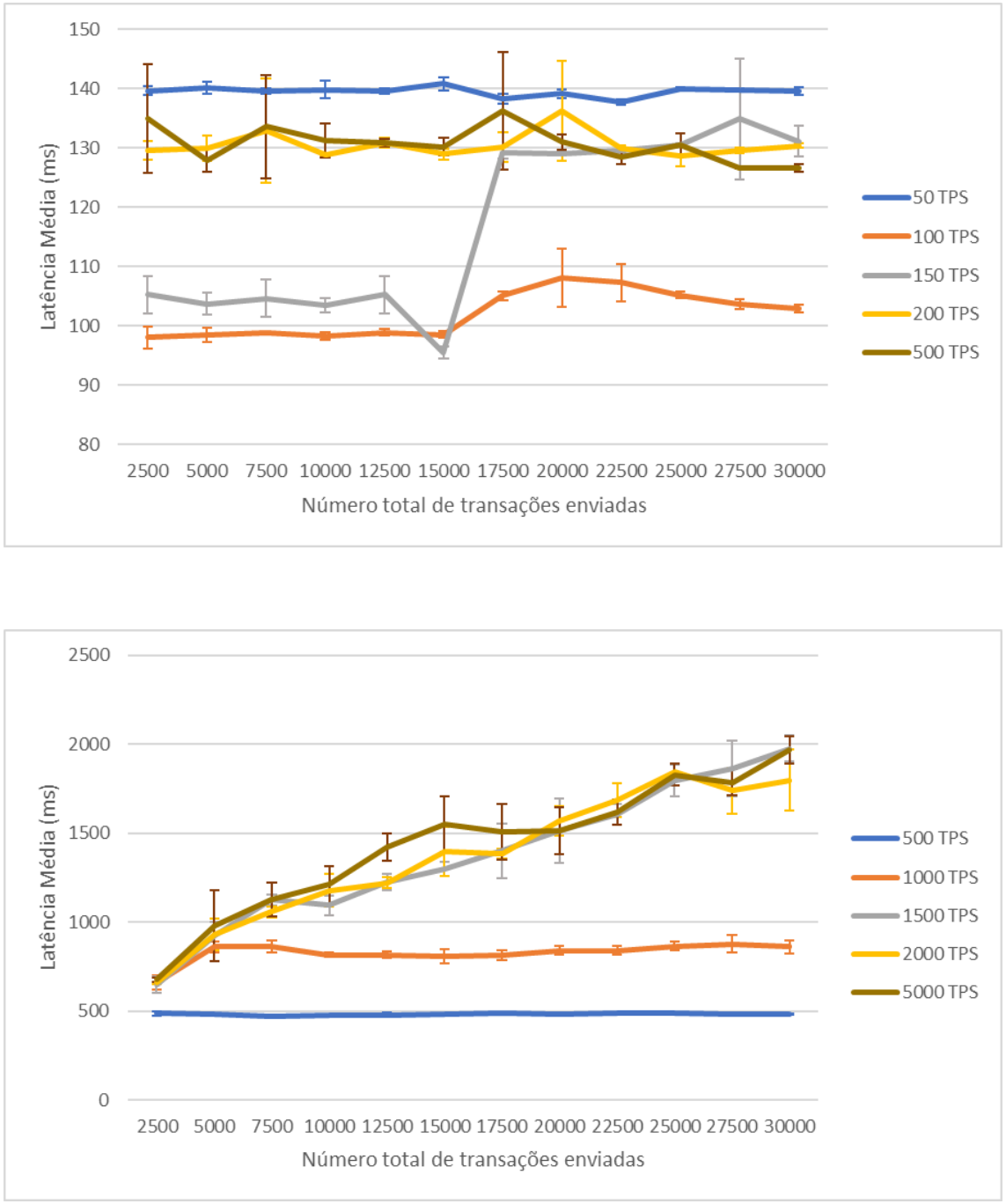


Figura 6.8: T3 - Representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric, Fundo: FISCO BCOS

Observando os gráficos acima é possível verificar que os intervalos de confiança são maiores quanto mais elevado é o *send rate* imposto. Ou seja, para o *send rate* de 500 TPS na *Fabric* e 5000 TPS na FISCO BCOS, os intervalos são maiores. Isto indica uma grande variação nos valores recolhidos.

Na figura 6.9 é apresentado o comportamento do *throughput* em função do *send rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Hyperledger Fabric* e o segundo à FISCO BCOS.

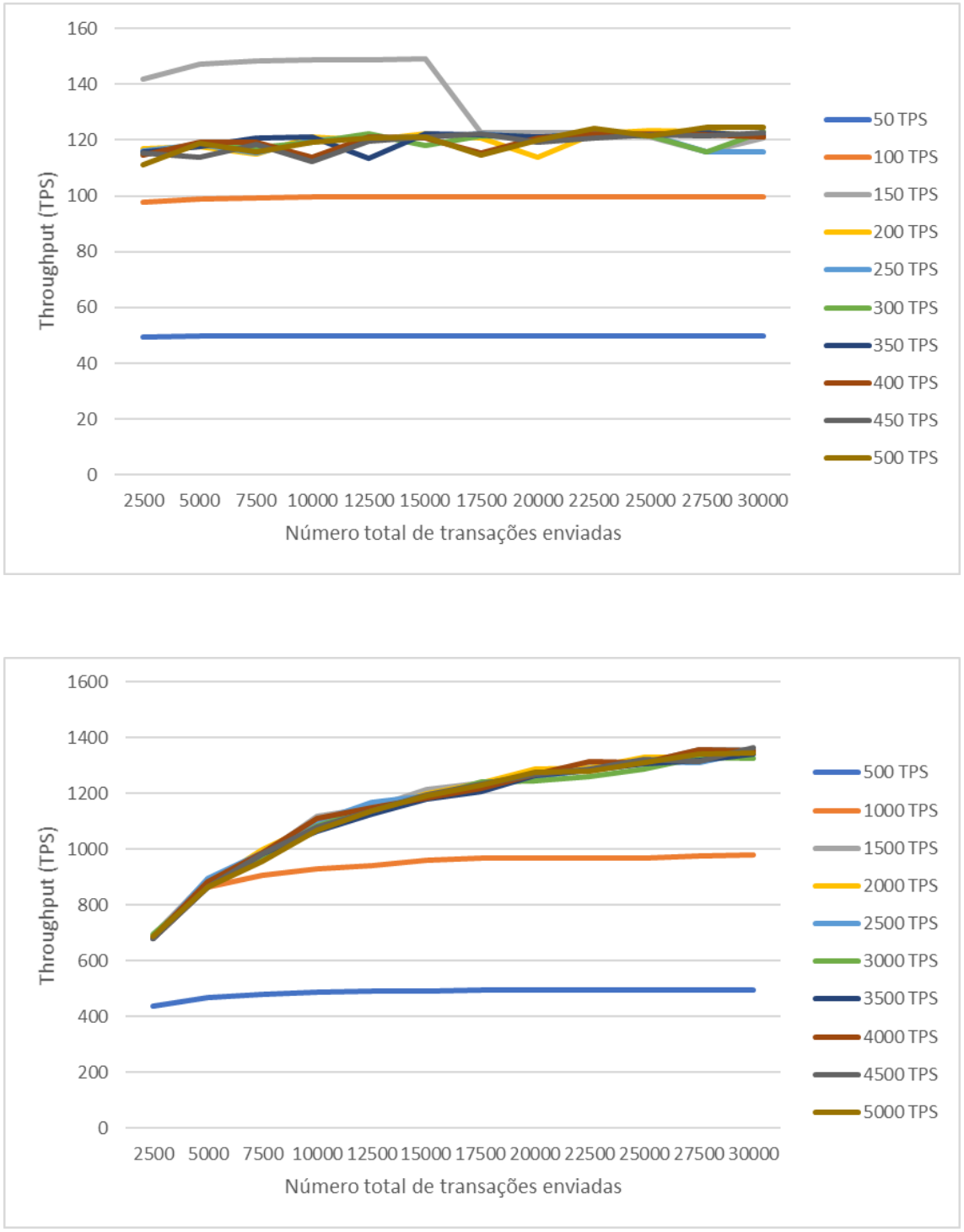


Figura 6.9: T3 - Comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric, Fundo: FISCO BCOS

Analisando os gráficos e tendo mais uma vez em consideração que os valores apresentados nos eixos são distintos em cada um deles, verifica-se que a plataforma FISCO BCOS atinge valores de *throughput* muito superiores aos alcançados pela *Hyperledger Fabric*. Em ambas as plataformas é notório um momento em que o *throughput* não consegue suportar o *send rate*. No caso da *Fabric*, este surge a partir das 100 TPS. No caso da FISCO BCOS, este momento dá-se a partir das 1000 TPS. A plataforma *Fabric* atinge um *throughput* constante por volta das 120 TPS, a partir do *send rate* de 200 TPS.

De seguida, na figura 6.10, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

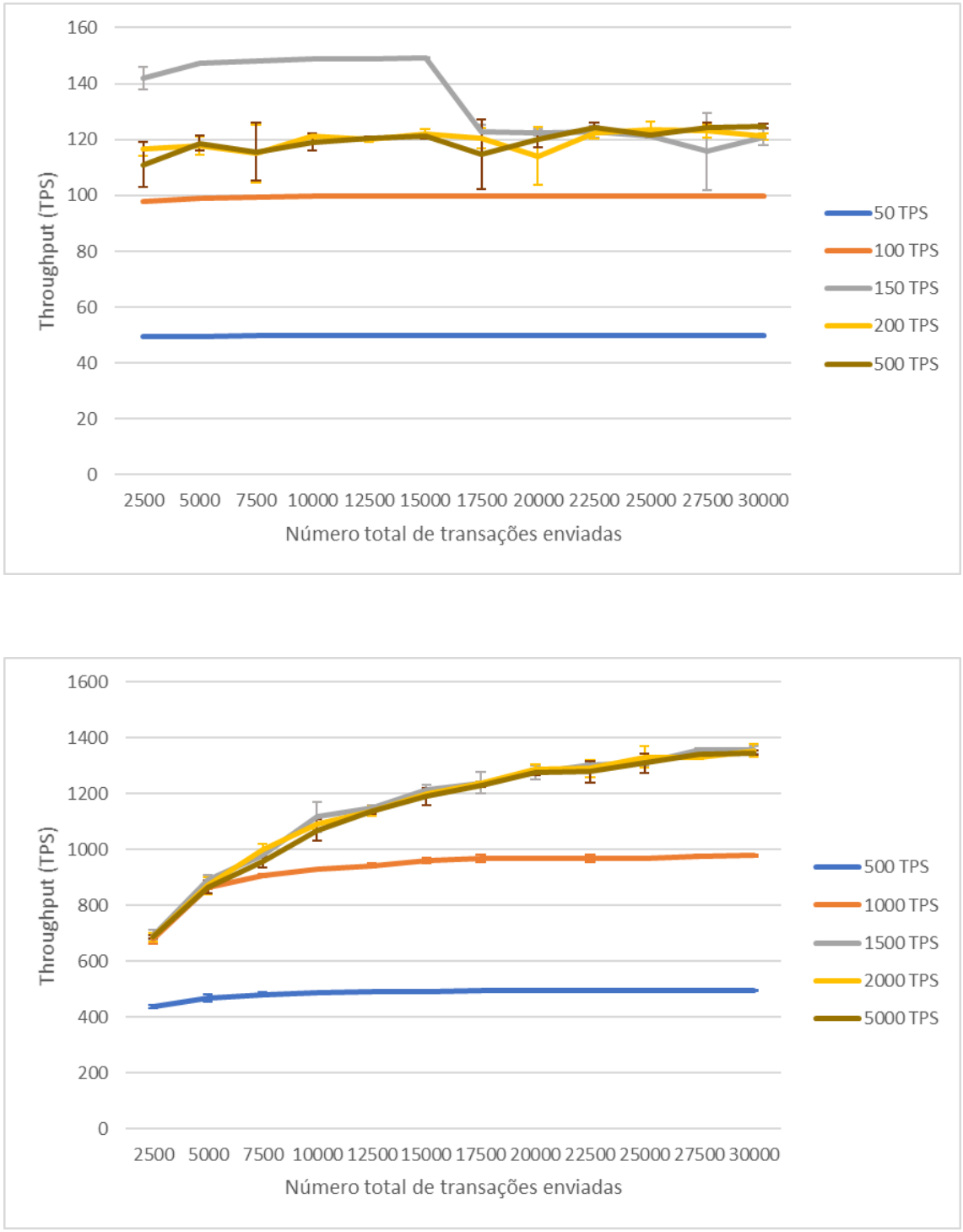


Figura 6.10: T3 - Representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric, Fundo: FISCO BCOS

Analisando os gráficos acima, é observável que o *send rate* de 500 TPS e 5000 TPS, na *Fabric* e na FISCO BCOS, respetivamente, são os valores que apresentam intervalos de confiança maiores.

Em suma, com este testes foi possível verificar que a plataforma FISCO BCOS apresenta valores de *Throughput* muito superiores aos obtidos na *Fabric*. Em relação ao valores de latência média obtidos, a *Hyperledger Fabric* apresenta um comportamento mais uniforme que a FISCO BCOS.

6.5 T4 - Teste de consumo de recursos realizando operações de escrita na *blockchain*

O teste T4 tem como principal objetivo estudar e comparar o consumo de recursos das plataformas *Hyperledger Fabric* e FISCO BCOS quando sobre elas são exercidas operações de escrita na *blockchain*. Com este propósito, no caso da *Fabric*, foi utilizada uma função que simula a transferência de uma certa quantia de dinheiro de uma conta para outra. No caso da FISCO BCOS, recorreu-se a uma função que altera a quantia de dinheiro de uma conta. Desta forma, é possível analisar o comportamento de cada uma das plataformas, no entanto, não é possível fazer uma comparação direta entre as mesmas, por se tratarem de operações de escrita distintas.

Na figura 6.11 encontra-se representada a percentagem de CPU utilizada em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* e o segundo à FISCO BCOS.

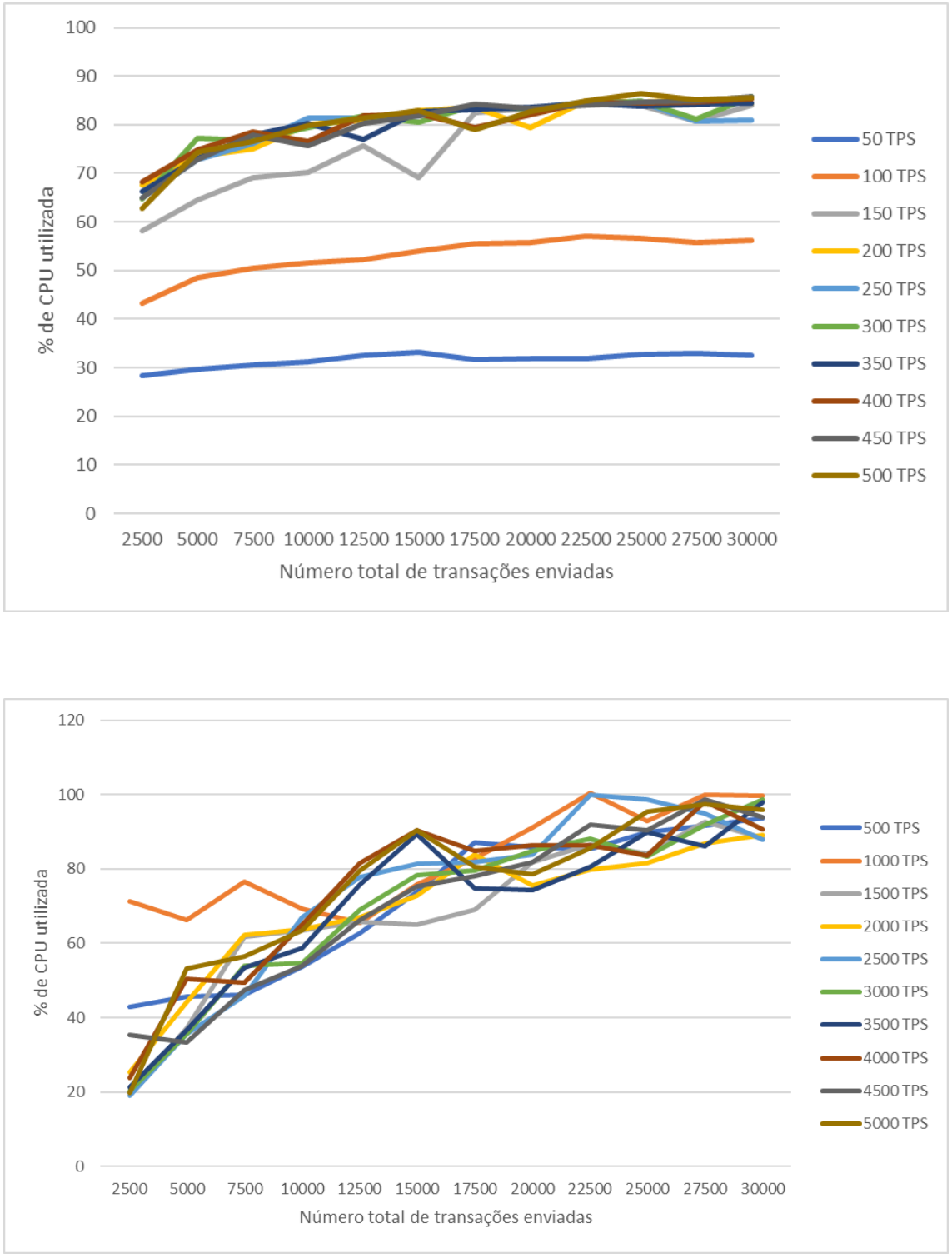


Figura 6.11: T4 - Percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric, Fundo: FISCO BCOS

Tendo em consideração os gráficos apresentados, verifica-se que a plataforma *Fabric* consome uma percentagem de recursos computacionais menor quanto menor for o *send rate*. Por outro lado, a FISCO BCOS apresenta um comportamento mais ou menos semelhante independentemente do *send rate*. Analisando os resultados, observa-se que a percentagem

de CPU utilizada pela FISCO BCOS é superior à utilizada pela *Fabric*, atingindo valores perto dos 100%, enquanto que a *Fabric* apenas chega perto dos 90%.

De seguida, na figura 6.12, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

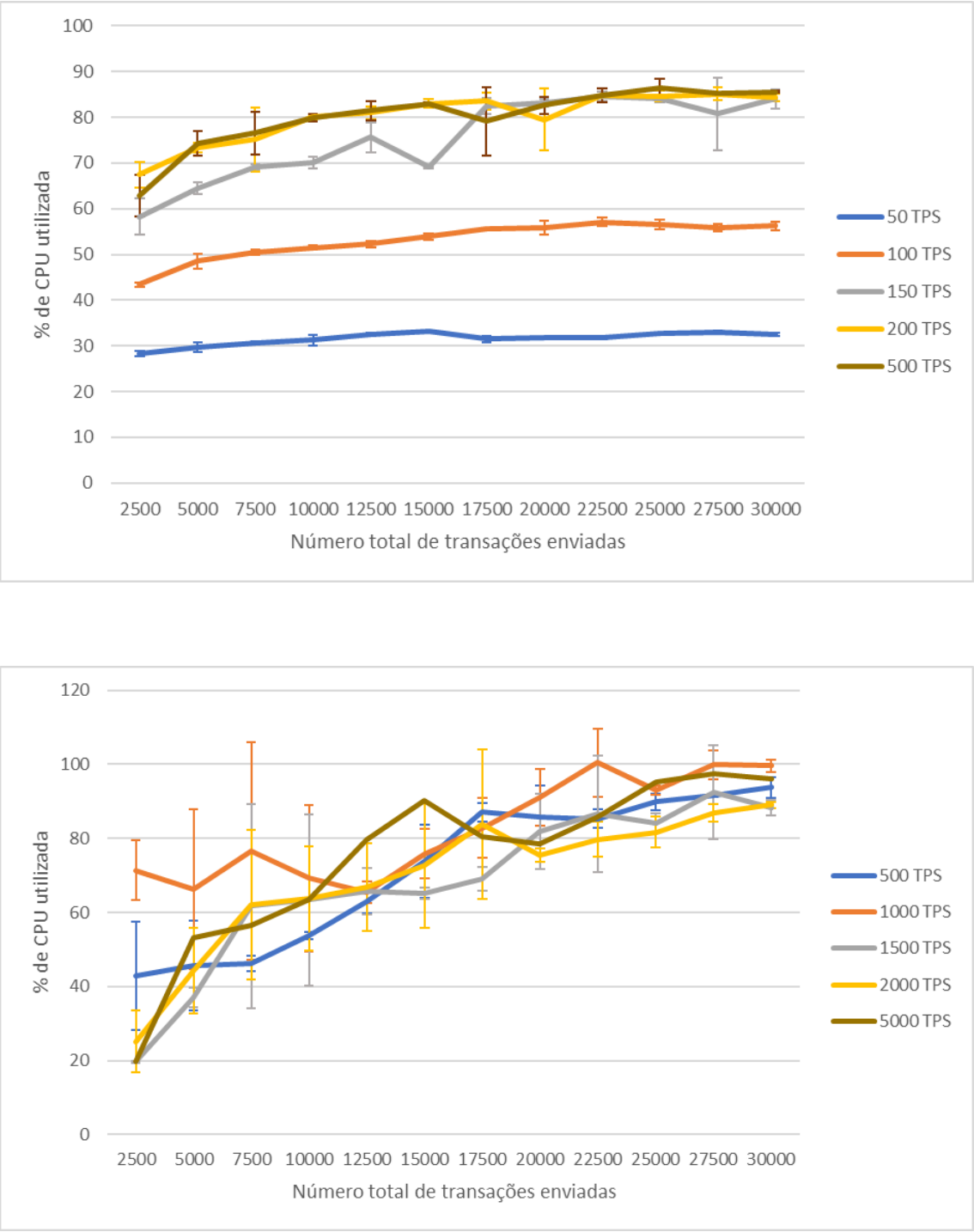


Figura 6.12: T4 - Representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric, Fundo: FISCO BCOS

Observando os gráficos acima é possível verificar que os intervalos de confiança para os valores obtidos da *Fabric* são pequenos, existindo raras excepções. No entanto, no caso da FISCO BCOS, os intervalos são grandes, o que indica uma grande variação nos valores recolhidos.

6.6 T5 - Teste à utilização do mecanismo de consenso *Raft*

O teste T5 tem como principal objetivo estudar e comparar o comportamento da plataforma *Hyperledger Fabric* quando é ou não utilizado o mecanismo de consenso *Raft*. Nesta secção procede-se à avaliação do comportamento da plataforma *Fabric* quando esta é submetida a operações de escrita na *blockchain*. Para a realização deste teste, foi utilizada uma função que simula a transferência de uma certa quantia de dinheiro de uma conta para outra, guardadas na *blockchain*.

Na figura 6.13 encontra-se representado o comportamento da latência média em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma Fabric quando não é utilizado o mecanismo de consenso *Raft* e o segundo à mesma plataforma mas com a utilização do *Raft*.

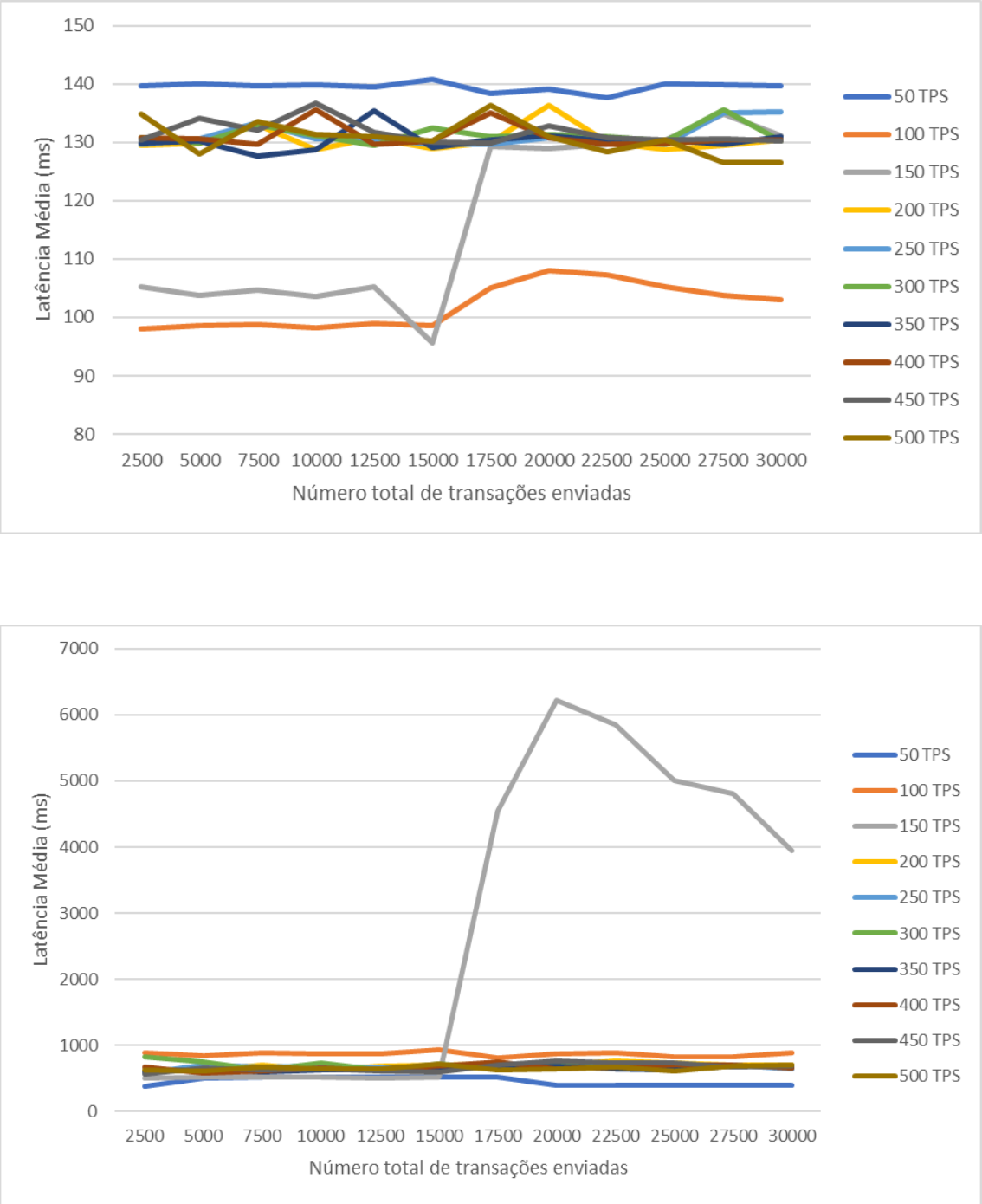


Figura 6.13: T5 - Comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Analisando os gráficos é possível verificar que, no primeiro caso, a *Fabric*, sem *Raft*, apresenta os maiores valores de latência média quando o *send rate* é de 50 TPS. Volta a existir a mudança abrupta no seu comportamento, nas 150 TPS, acima das 15000 transações enviadas. Nesse momento, a plataforma apresenta um comportamento semelhante àquele que acontece em *send rates* superiores a 150 TPS. No caso do segundo gráfico, é de notar que os valores nos eixos são bastante distintos do gráfico anterior. Na *Fabric*, com *Raft*, a latência média apresenta valores muito superiores aos representados quando

não é utilizado o mecanismo de consenso. Mais uma vez, nas 150 TPS, a partir das 15000 transações, há uma mudança abrupta no comportamento da latência média.

De seguida, na figura 6.14, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

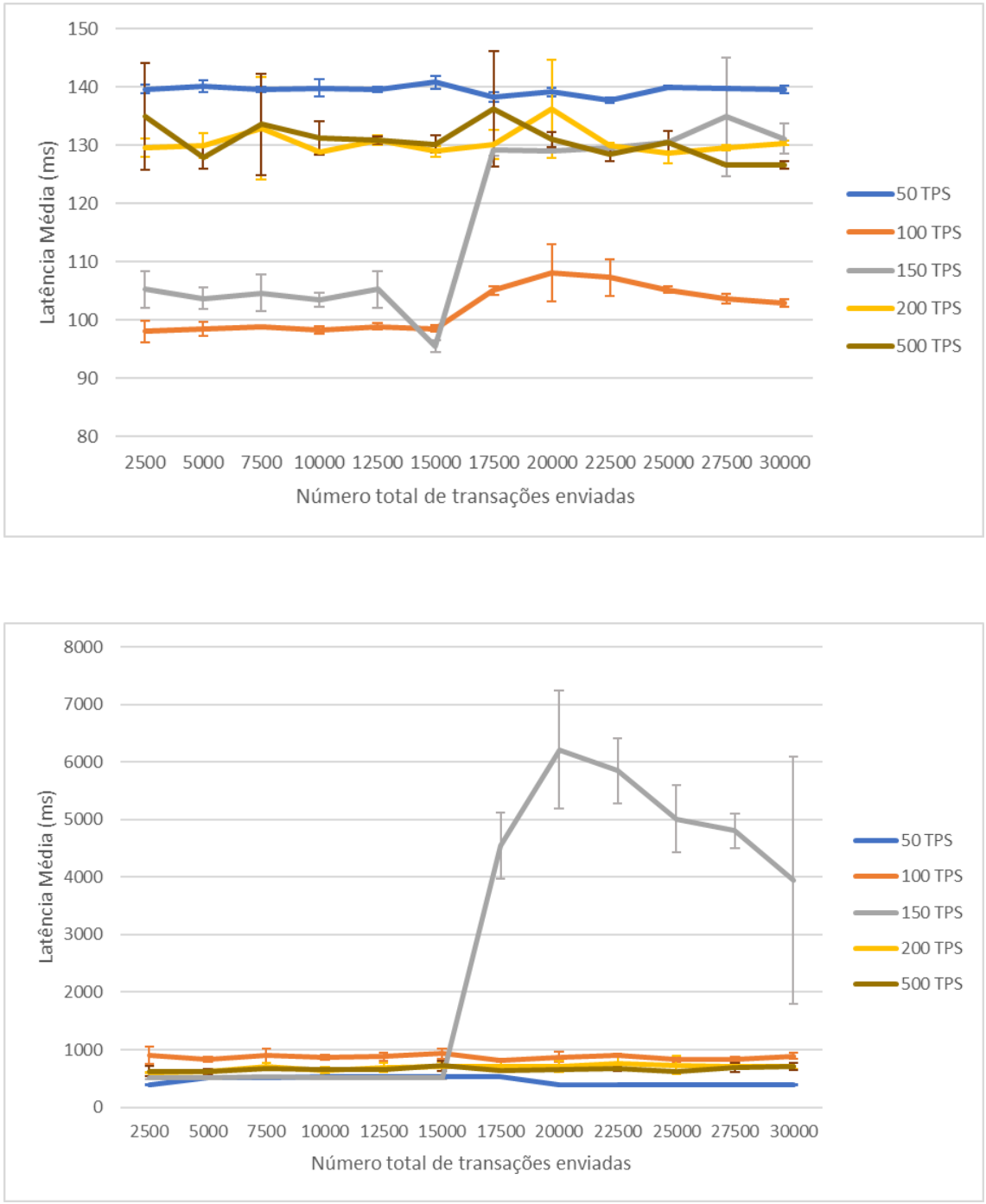


Figura 6.14: T5 - Representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Os gráficos acima correspondem à representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* sem *Raft* e com *Raft*, respetivamente. Observando os gráficos acima é possível verificar que os intervalos de confiança para os valores obtidos da *Fabric*, com *Raft*, são geralmente pequenos. O *send rate* de 150 TPS, a partir das 15000 transações, é o momento em que os intervalos de confiança são maiores, ou seja, há mais variação nos valores. No entanto, no caso da *Fabric*, sem *Raft*, os intervalos de confiança já são maiores, o que indica uma grande variação nos valores recolhidos. Este aumento nos intervalos de confiança verifica-se ser maior, quanto maior é o *send rate* que está a ser estudado.

Na figura 6.15 é apresentado o comportamento do *throughput* em função do *send rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma Fabric quando não é utilizado o mecanismo de consenso *Raft* e o segundo à mesma plataforma mas com a utilização do *Raft*.

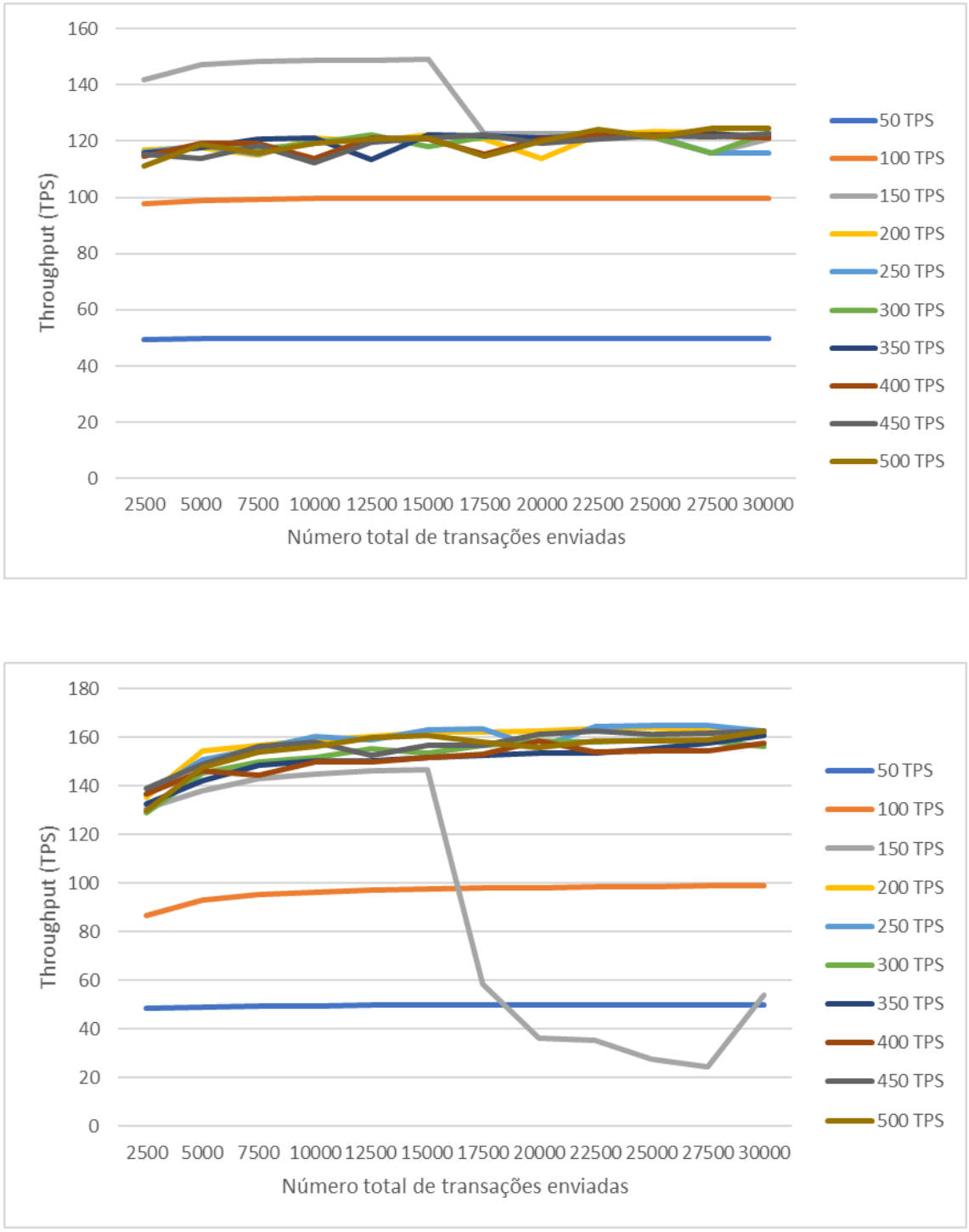
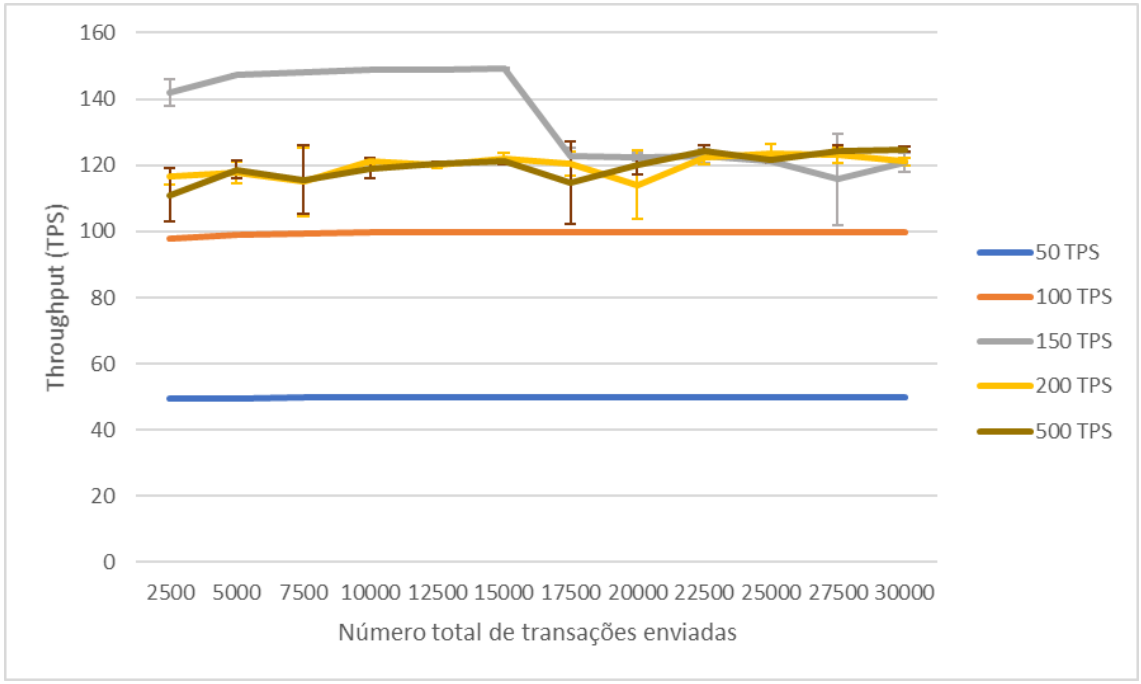


Figura 6.15: T5 - Comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Fazendo uma análise aos gráficos apresentados, é possível observar que o gráfico correspondente à *Fabric*, sem *Raft*, apresenta valores constantes para os *send rates* de 50 TPS e 100 TPS, voltando a existir uma mudança nos 150 TPS. O comportamento do *throughput* torna-se mais ou menos constante por volta dos 120 TPS. Em relação à *Fabric*, com *Raft*, os *send rates* de 50 TPS e 100 TPS também são mais ou menos constantes nesses valores. Existe uma mudança abrupta no comportamento quando o *send rate* é de 150 TPS. O *throughput* atinge um nível mais ou menos estável por volta dos 160 TPS.

De seguida, na figura 6.16, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.



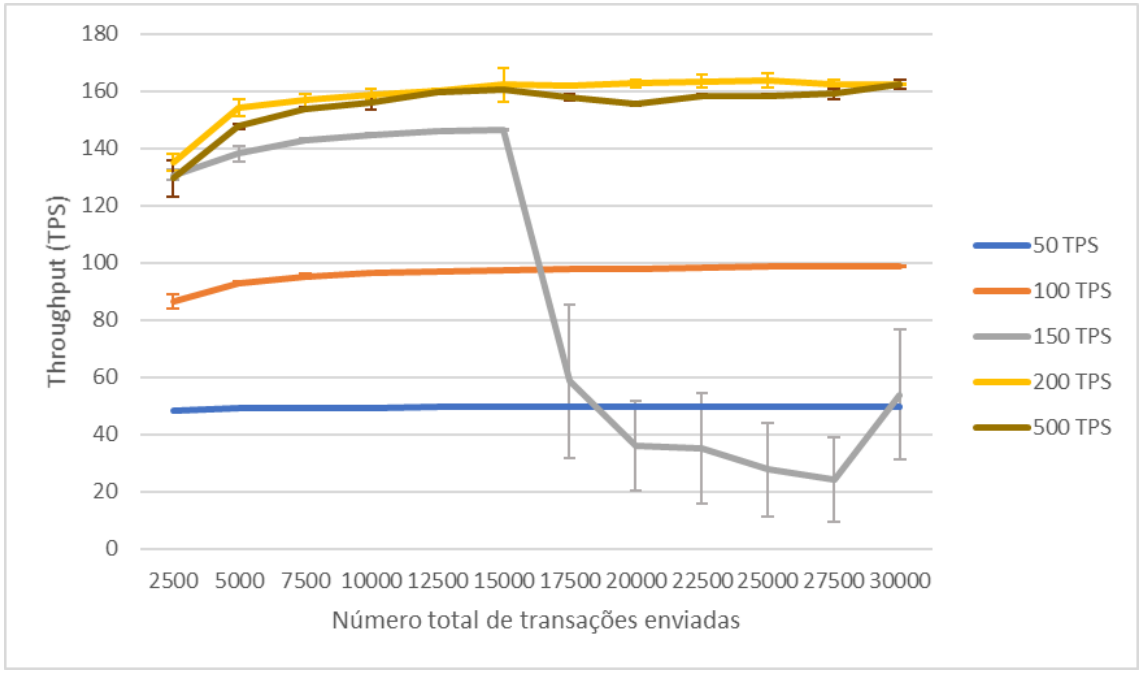
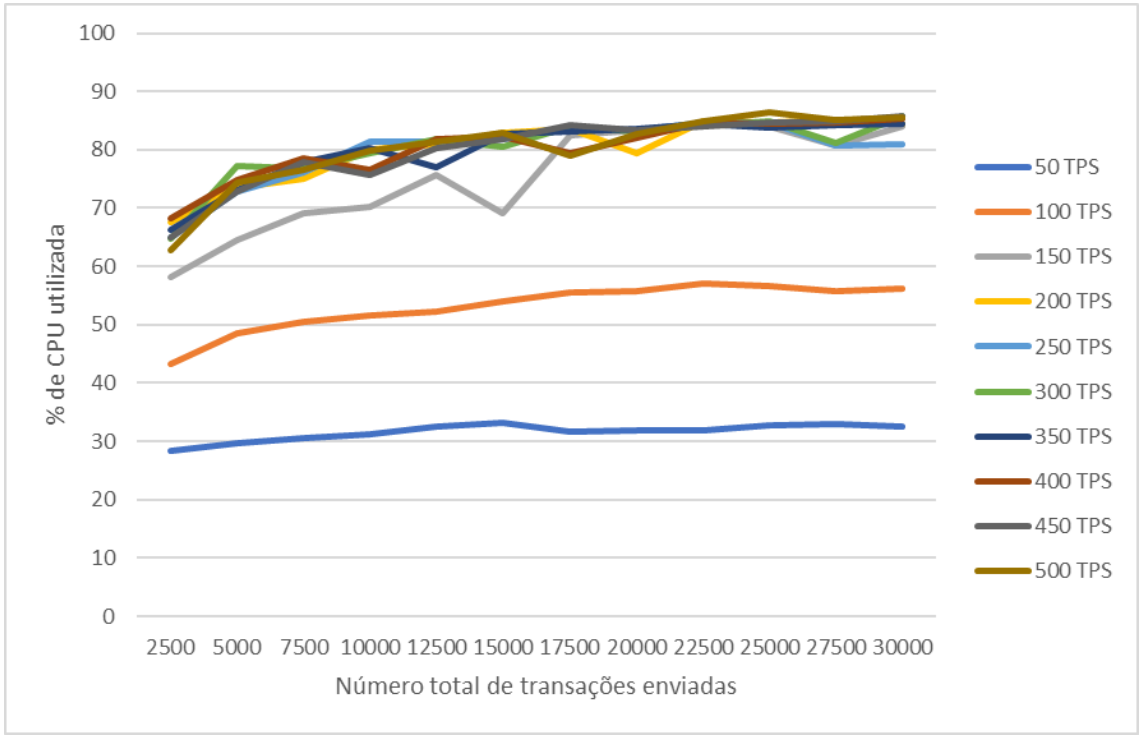


Figura 6.16: T5 - Representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Os gráficos acima correspondem à representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* sem *Raft* e com *Raft*, respetivamente. Observando os gráficos acima é possível verificar que os intervalos de confiança para os valores obtidos da *Fabric*, em ambos os casos, são geralmente pequenos. O *send rate* de 150 TPS, a partir das 15000 transações, é

o momento em que os intervalos de confiança são maiores, no caso da utilização do *Raft*, ou seja, há mais variação nos valores. No entanto, no caso da *Fabric*, sem *Raft*, os intervalos de confiança são maiores quando o *send rate* é de 500 TPS.

Na figura 6.17 encontra-se representada a percentagem de CPU utilizada em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma Fabric quando não é utilizado o mecanismo de consenso *Raft* e o segundo à mesma plataforma mas com a utilização do *Raft*.



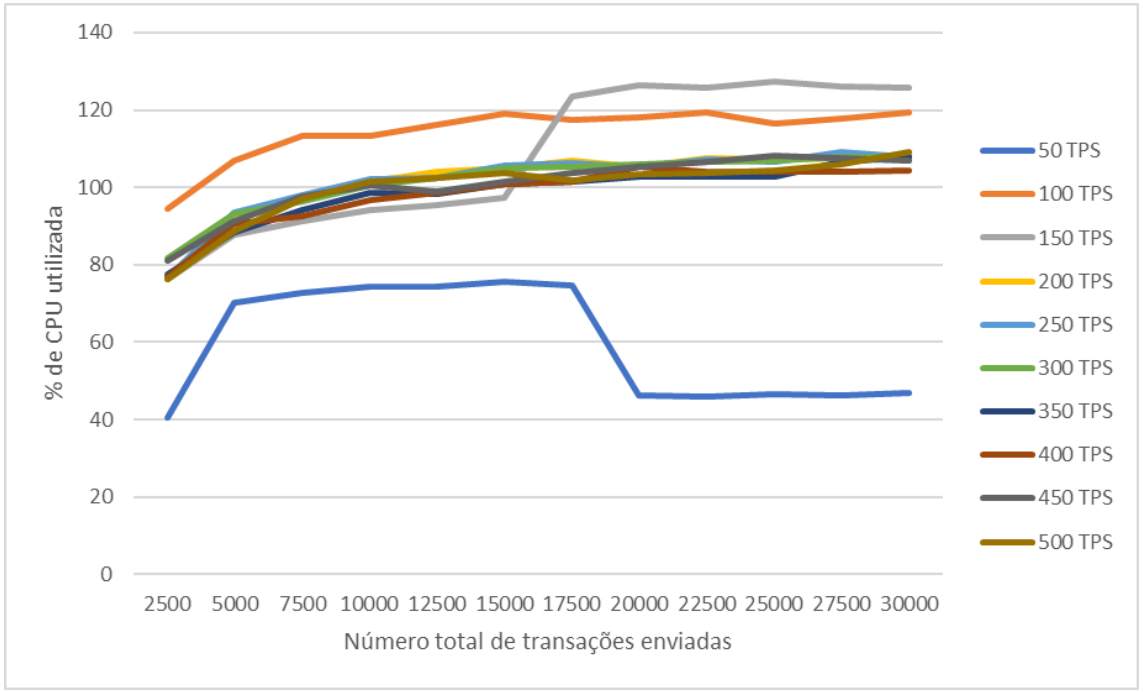


Figura 6.17: T5 - Percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Observando os gráficos é notório que a percentagem de CPU utilizada quando se utiliza o mecanismo de consenso *Raft* é bastante superior à percentagem consumida quando este não é utilizado. Quando o mecanismo de consenso é utilizado, a percentagem de CPU utilizada alcança valores acima dos 100%. No caso da *Fabric*, sem *Raft*, a percentagem de CPU utilizada anda por volta dos 90%, quando submetida a *send rates* altos. É possível verificar que o consumo de CPU, no caso do *Raft*, é mais instável para *send rates* mais baixos.

De seguida, na figura 6.18, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

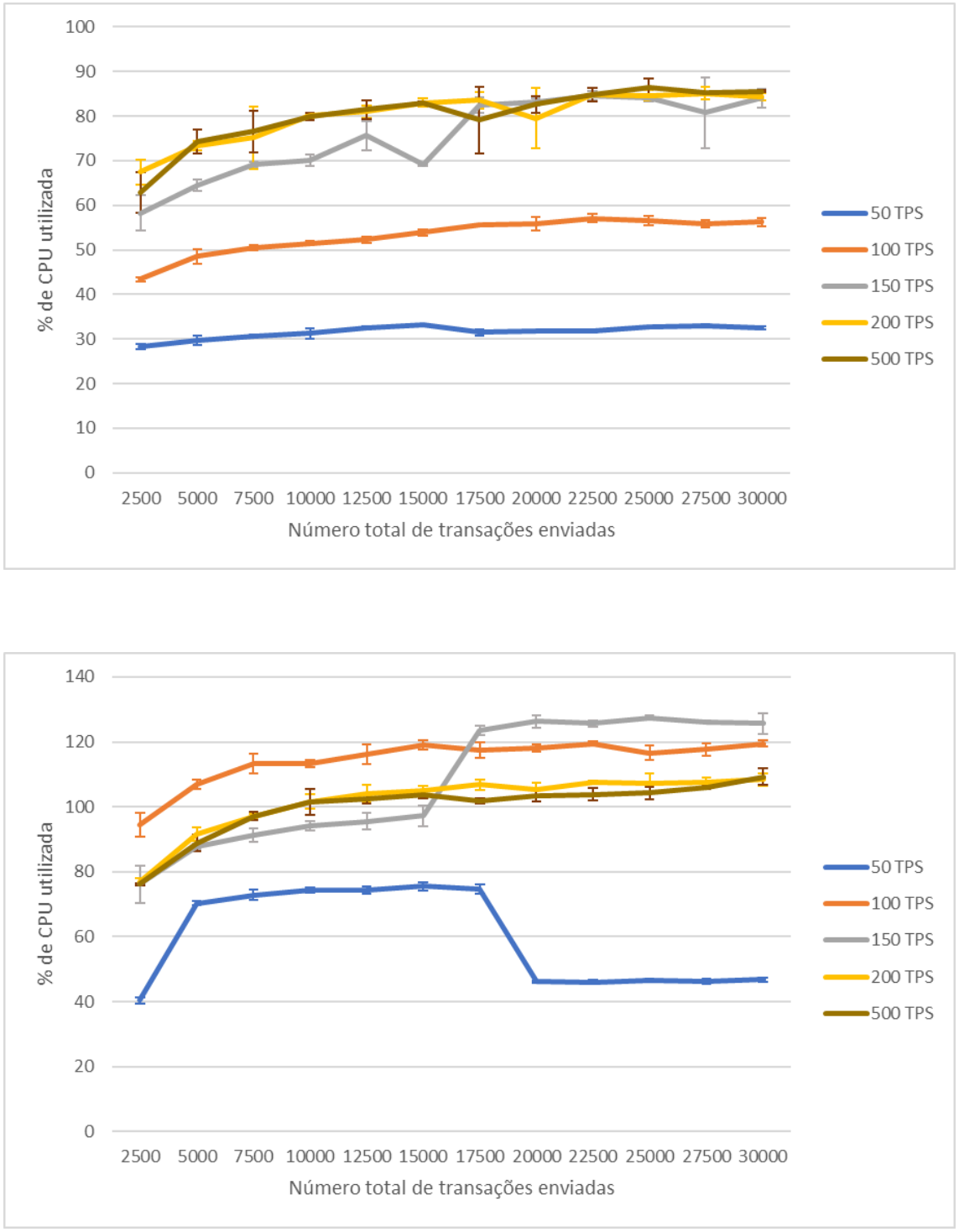


Figura 6.18: T5 - Representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Os gráficos acima correspondem à representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* sem *Raft* e com *Raft*, respetivamente. Analisando os gráficos é possível verificar que, em ambos, os intervalos de confiança para os valores obtidos são pequenos.

Na figura 6.19 encontra-se representado o número de transações que são bem sucedidas em função do *send rate* e do número total de transações enviadas. O primeiro gráfico diz respeito à plataforma Fabric quando não é utilizado o mecanismo de consenso *Raft* e o segundo à mesma plataforma mas com a utilização do *Raft*.

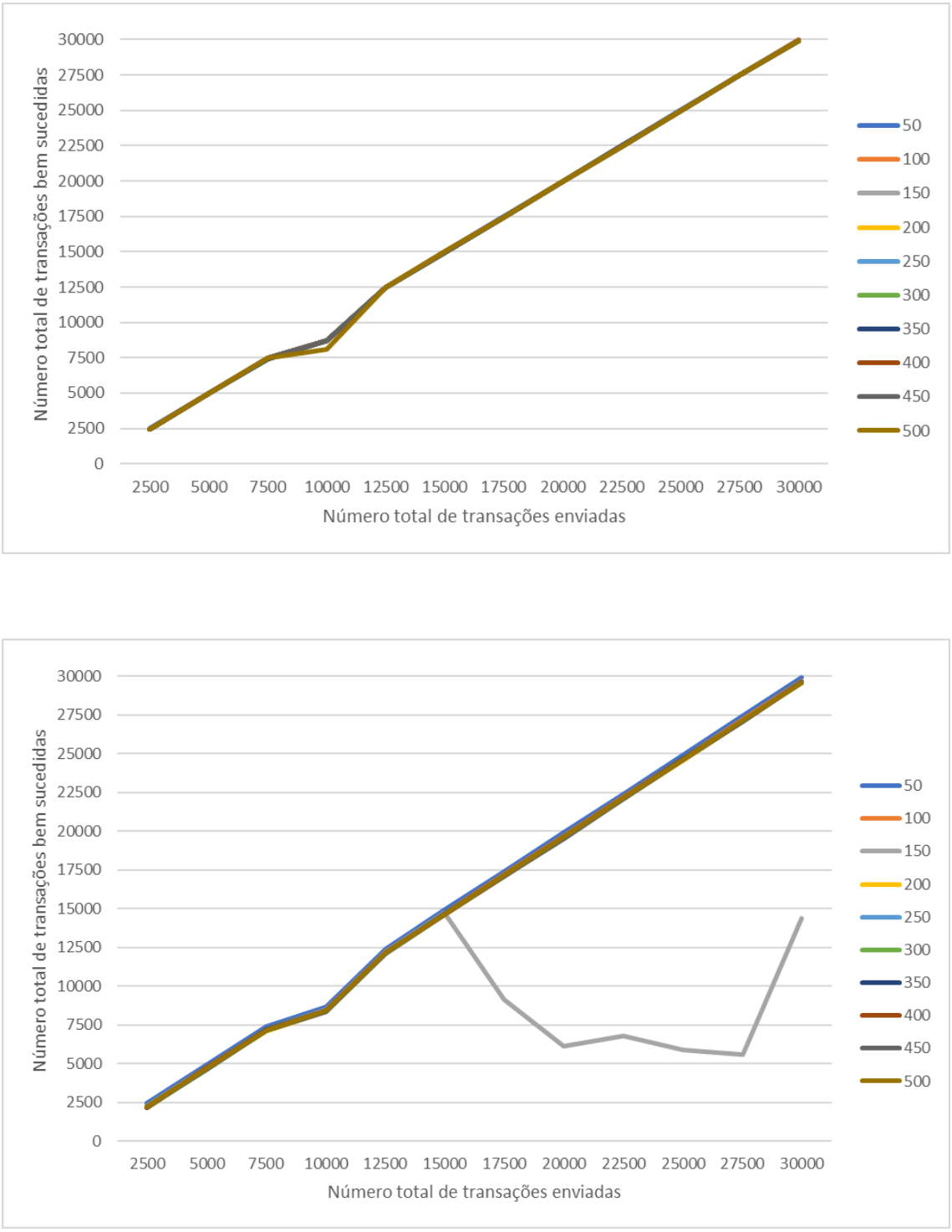


Figura 6.19: T5 - Número de transações bem sucedidas em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - Sem *Raft*, Fundo: Fabric - Com *Raft*

Observando os gráficos é possível observar que o número de transações bem sucedidas quando é utilizado o mecanismo *Raft* é inferior à conseguida quando este não é usado, apesar de a diferença ser muito baixa. Contudo, à medida que o número total de transações enviadas aumenta, esta diferença deixa de ser tão notória, passando a atingir valores muito semelhantes. A diferença mais relevante passa pelo *send rate* de 150 TPS. Este é o único

que afecta diretamente o funcionamento do sistema, como já tinha sido verificado em gráficos anteriores. Pode ser um momento em que o sistema entra em sobrecarga.

Resumindo, o comportamento da plataforma *Hyperledger Fabric* é afectado de forma notória, quando utilizado o *Raft*, no que diz respeito à latência, ao consumo de CPU e à taxa de sucesso. Assim, pode afirmar-se que a utilização do *Raft* torna o processo muito mais demorado e pesado computacionalmente. Com a utilização do *Raft* também foi possível observar uma anormalidade no comportamento da plataforma quando o *send rate* é de 150 TPS.

6.7 T6 - Teste à utilização de diferentes bases de dados

O teste T6 tem como principal objetivo estudar e comparar o comportamento da plataforma *Hyperledger Fabric* quando esta utiliza *LevelDB* ou *CouchDB*. De forma a proceder a esta avaliação, foram efetuadas operações de leitura e escrita na *blockchain*.

Nas secções 6.7.1 e 6.7.2 são apresentados e analisados os resultados obtidos para a execução de operações de leitura e escrita na *blockchain*, respetivamente.

6.7.1 Operações de Leitura na *blockchain*

Nesta secção pretende-se analisar o comportamento da plataforma *Fabric* quando esta é submetida a operações de leitura na *blockchain*. Para a realização deste teste, foi utilizada uma função que lê os dados guardados na *blockchain*. Foi estudada a plataforma *Fabric* quando esta utiliza *LevelDB* ou *CouchDB*.

Na figura 6.20 encontra-se representado o comportamento da latência média em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

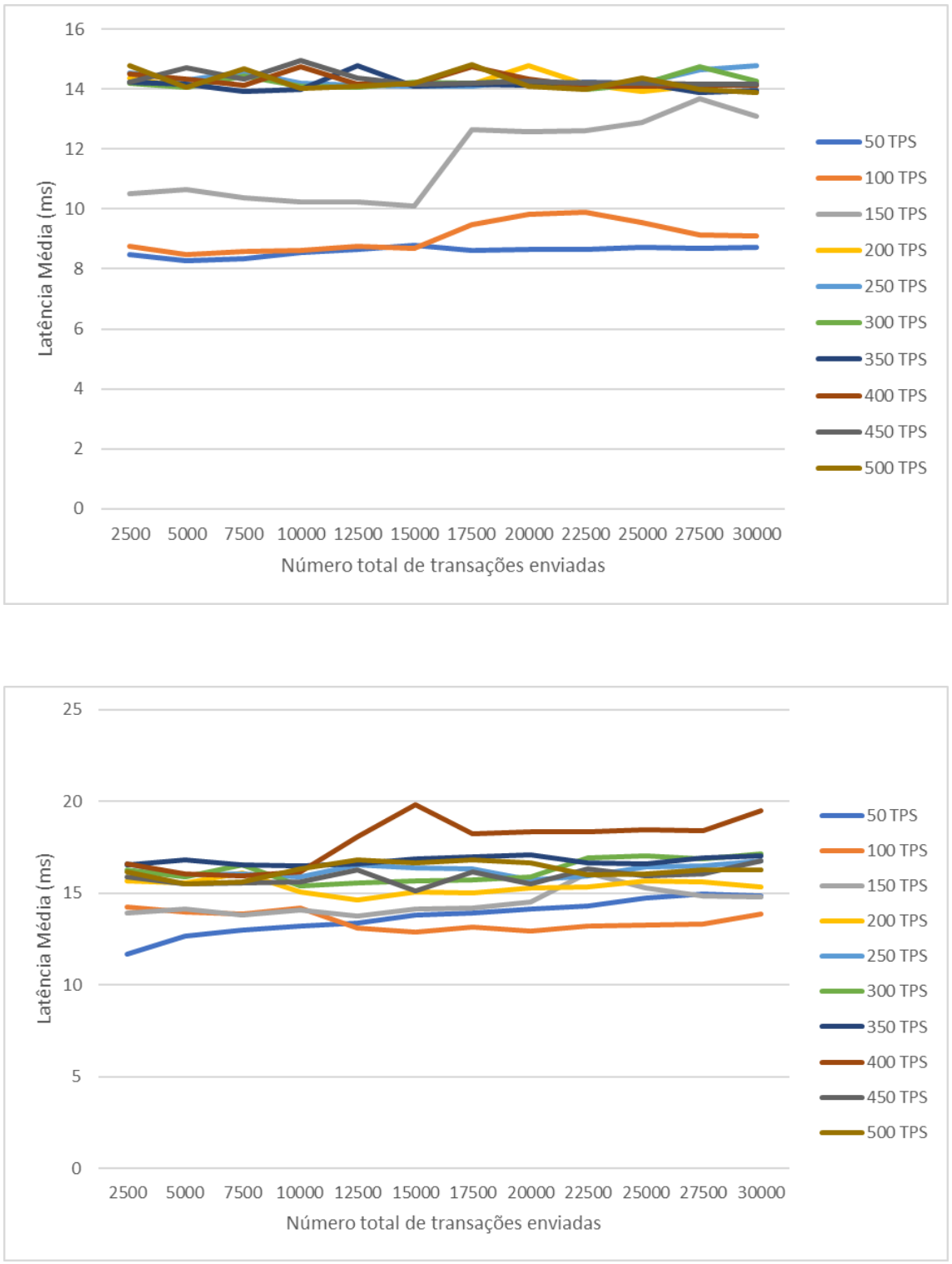


Figura 6.20: T6 - Comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

A partir dos gráficos, é possível observar que, em ambas as plataformas, os valores de latência média não ultrapassam os 20 milissegundos. É também importante reparar nos valores dos eixos, visto que apresentam escalas diferentes. Em relação à *Fabric* com *LevelDB*, quando submetida a *Send Rates* entre as 50 e as 150 TPS, esta demora menos tempo a responder aos pedidos de transações, apresentando assim valores de latência média mais baixos. Contudo, é de salientar o comportamento desta plataforma quando atinge as 15000 transações enviadas, com um *send rate* de 150 TPS. A partir deste valor, a plataforma possivelmente entra em sobrecarga e aumenta, de forma significativa, o tempo de resposta aos pedidos que lhe são efetuados. A partir das 150 TPS, esta plataforma estabiliza, sendo que, mesmo alterando o *send rate* e o número de transações que são enviadas para a *blockchain*, a sua latência média mantém-se mais ou menos constante, entre os 14 e os 15 milissegundos, como já tinha sido mostrado anteriormente. No que diz respeito à *Fabric* com *CouchDB*, é notório que para *send rates* mais elevados, a latência média vai também aumentando. Contudo, para cada *send rate*, esta apresenta valores de latência média mais ou menos constantes.

De seguida, na figura 6.21, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

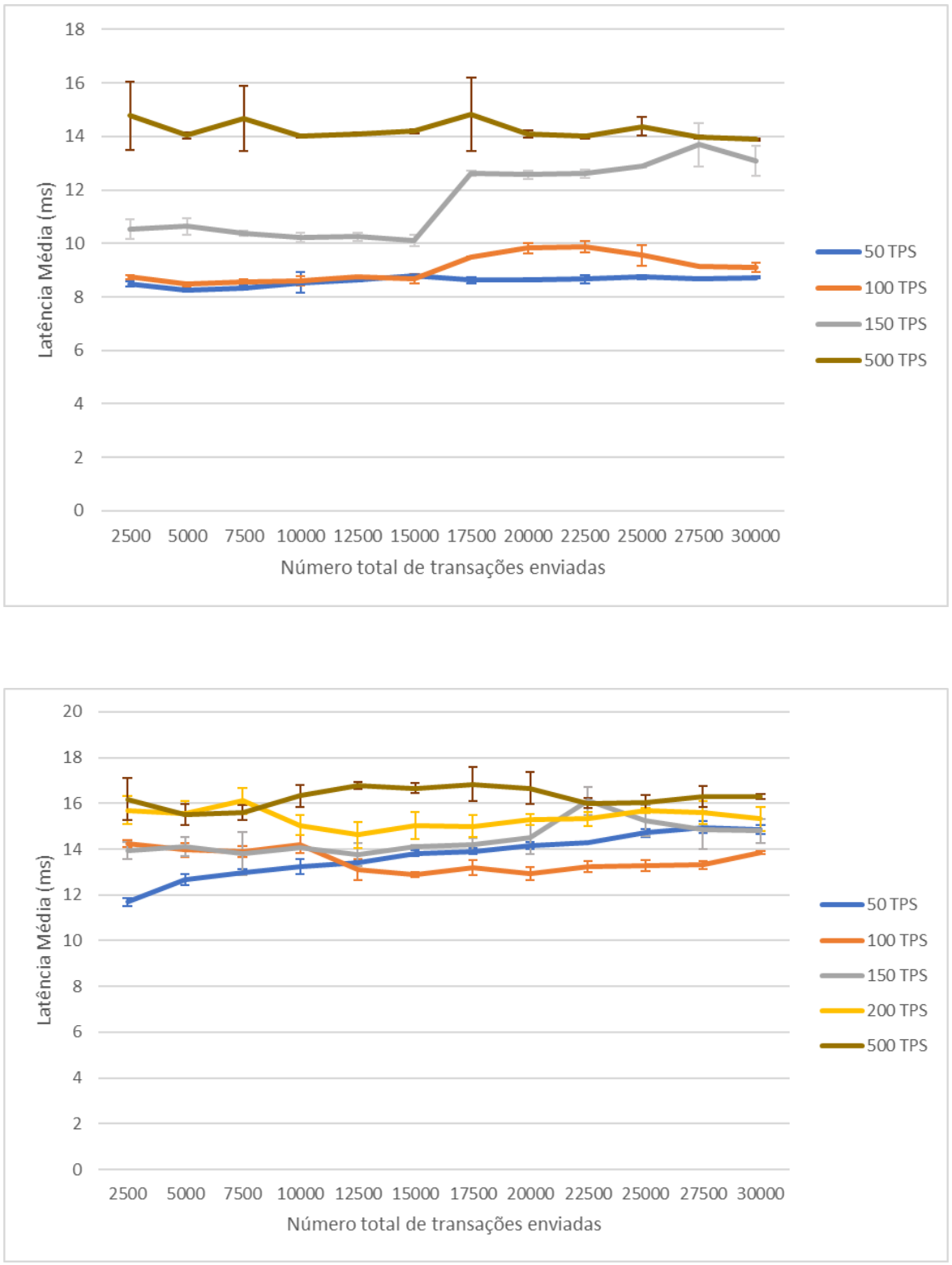


Figura 6.21: T6 - Representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Os gráficos acima correspondem à representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* com *LevelDB* e com *CouchDB*, respetivamente. Com a representação dos intervalos de confiança, é possível verificar que estes são menores na plataforma *Hyperledger Fabric* com *LevelDB*, com a excepção dos resultados obtidos para um *send rate* de 500 TPS. Utilizando *CouchDB*, os valores de intervalo de confiança são pequenos, contudo são visíveis em todos os valores de *send rate*.

Na figura 6.22 é apresentado o comportamento do *throughput* em função do *send rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

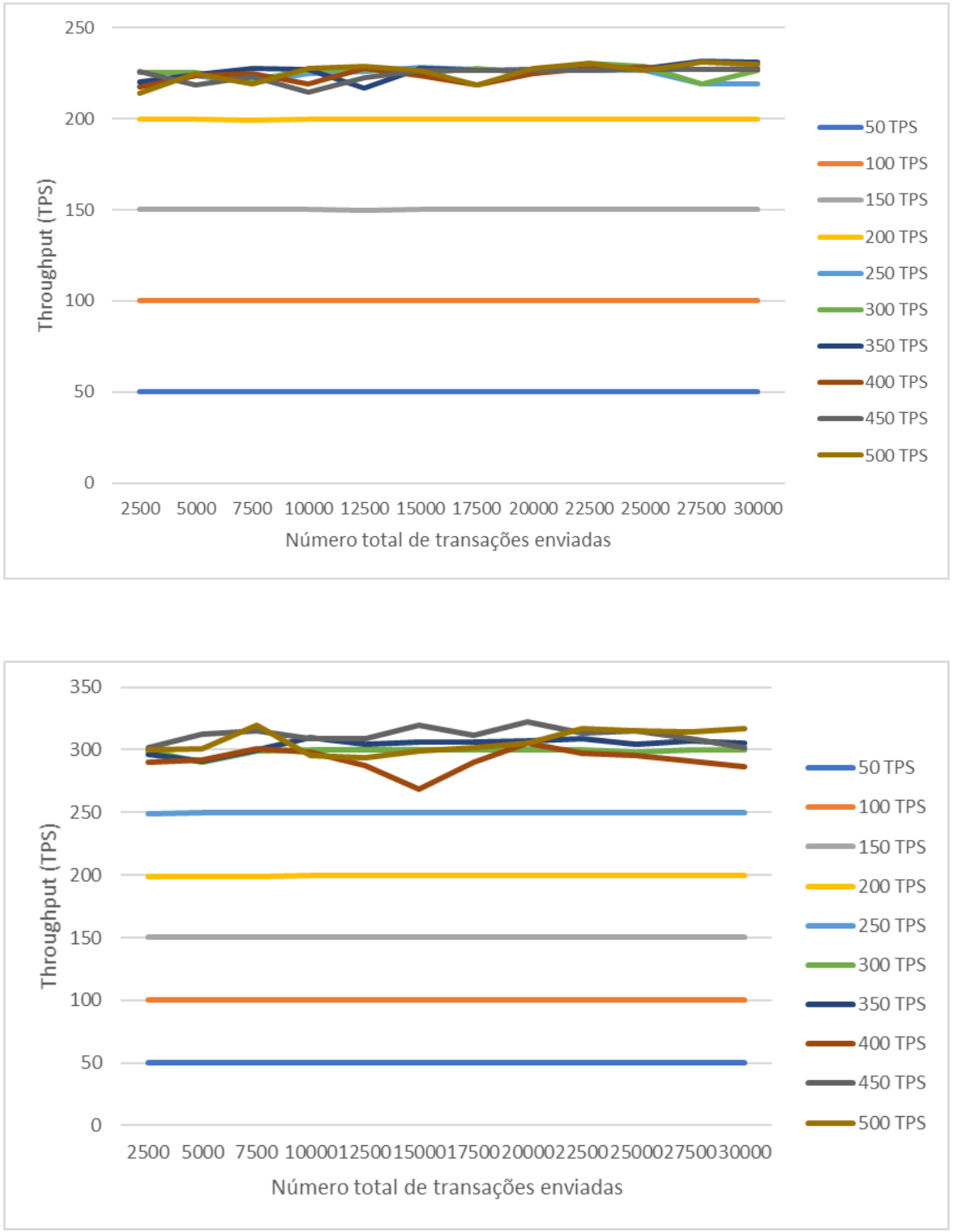


Figura 6.22: T6 - Comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Analisando os gráficos e tendo mais uma vez em consideração que os valores apresentados nos eixos são distintos em cada um deles, verifica-se que a plataforma *Fabric* com *CouchDB* atinge valores de *throughput* superiores aos alcançados pela *Fabric* com *LevelDB*. Em ambas as plataformas é notório um momento em que o *throughput* não consegue acompanhar o *send rate* que é imposto. No caso da *Fabric* com *LevelDB*, este surge a partir das 200 TPS. No caso da *CouchDB*, a partir das 250 TPS. A plataforma *Fabric* com *LevelDB* atinge um *throughput* constante por volta das 220 TPS, a partir do *send rate* de 200 TPS. A *Fabric* com *CouchDB* apresenta um valor de *throughput* mais ou menos constante, por volta das 300 TPS, a partir do *send rate* de 250 TPS.

De seguida, na figura 6.23, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

Figura 6.23: T6 - Representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Os gráficos acima correspondem à representação do *Throughput* em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* com *LevelDB* e com *CouchDB*, respetivamente. Com a representação dos intervalos de confiança, de forma semelhante ao que acontece nos gráficos de *send rate*, os intervalos de confiança não são significantes na maior parte dos valores obtidos, em ambos os casos. O *send rate* de 500 TPS são os valores que apresentam intervalos de confiança maiores.

Na figura 6.24 encontra-se representada a percentagem de CPU utilizada em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

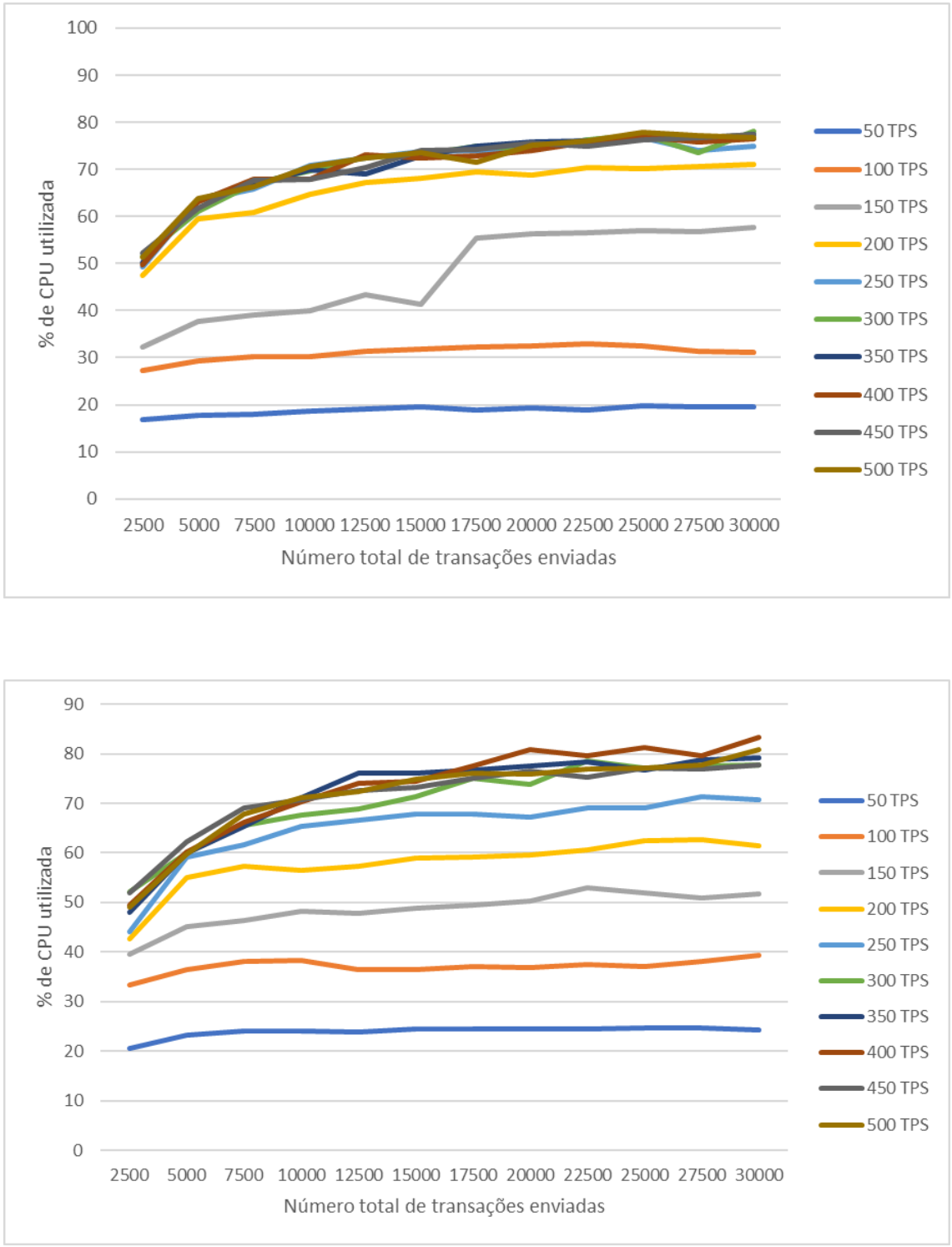


Figura 6.24: T6 - Percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Observando os gráficos é possível verificar que em ambos os casos a percentagem máxima de CPU obtida anda perto dos 80%. É ainda de referir que quando mais baixo o *send rate*, mais baixa é a percentagem de CPU que está a ser utilizada. Apenas a partir do *send rate* de 200 TPS é que os valores se vão tornando mais próximos e constantes.

De seguida, na figura 6.25, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

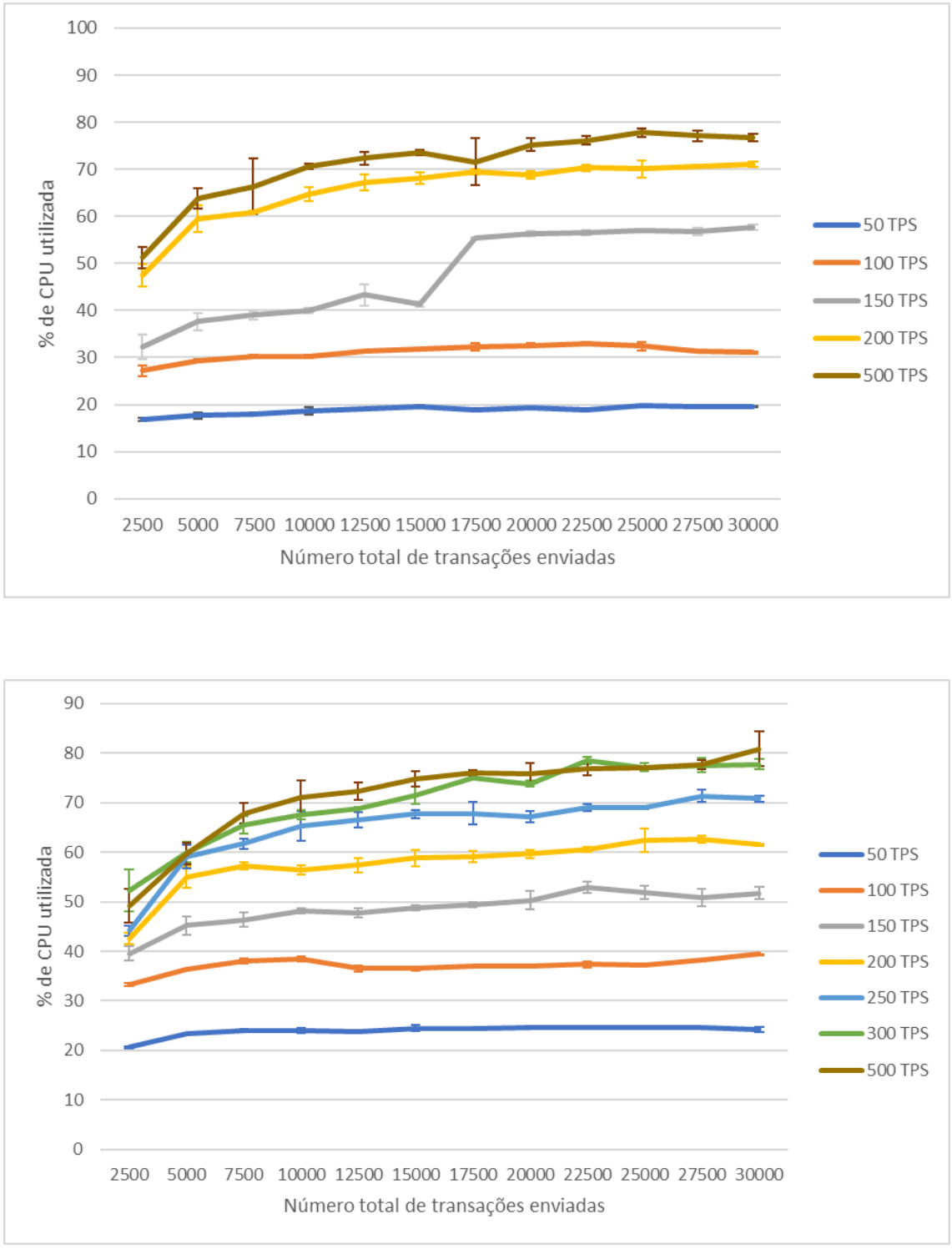


Figura 6.25: T6 - Representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Os gráficos acima correspondem à representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* com *LevelDB* e com *CouchDB*, respetivamente. Analisando os gráficos é possível verificar que, em ambos, os intervalos de confiança para os valores obtidos são pequenos.

Em suma, com este testes foi possível verificar que a plataforma *Fabric* com *CouchDB* apresenta valores de *Throughput* superiores aos obtidos na *Fabric* com *LevelDB*.

6.7.2 Operações de Escrita na *blockchain*

Nesta secção procede-se à avaliação do comportamento da plataforma Fabric quando esta é submetida a operações de escrita na *blockchain*. Para a realização deste teste, foi utilizada uma função que simula a transferência de uma certa quantia de dinheiro de uma conta para outra, guardadas na *blockchain*.

Na figura 6.26 encontra-se representado o comportamento da latência média em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

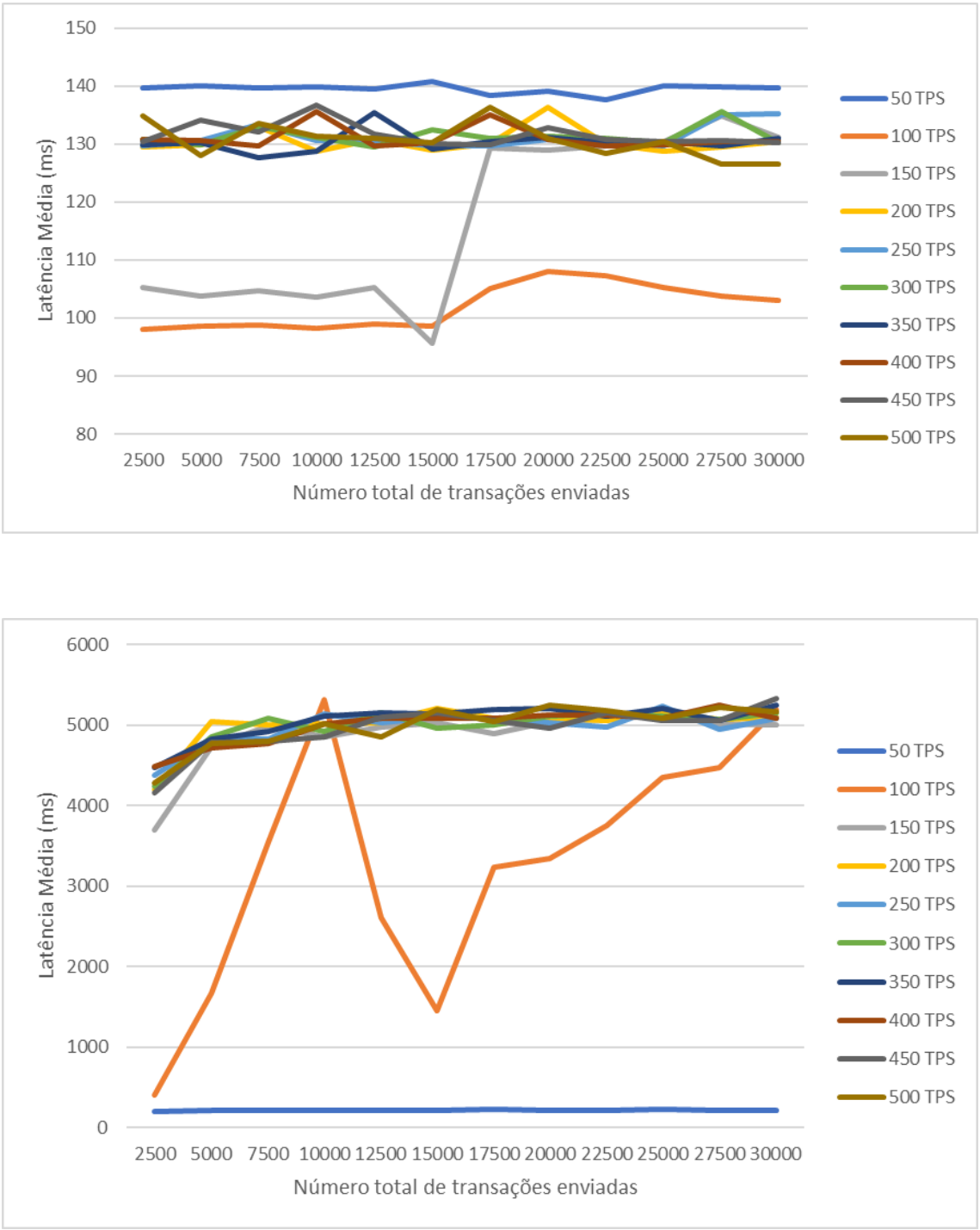


Figura 6.26: T6 - Comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Analisando os gráficos é possível verificar que, no primeiro caso, como já tinha sido visto anteriormente, a *Fabric*, com *LevelDB*, apresenta os maiores valores de latência média quando o *send rate* é de 50 TPS. Volta a existir a mudança abrupta no seu comportamento, nas 150 TPS, acima das 15000 transações enviadas. Neste momento, a plataforma apresenta um comportamento semelhante àquele que acontece em *send rates* superiores a 150 TPS. No caso do segundo gráfico, é de notar que os valores nos eixos são bastante distintos do gráfico anterior. Na *Fabric*, com *CouchDB*, a latência média apresenta valores muito superiores aos apresentados quando é utilizado *LevelDB*. Nas 100 TPS, acontecem vários momentos em que existem mudanças abruptas.

De seguida, na figura 6.27, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

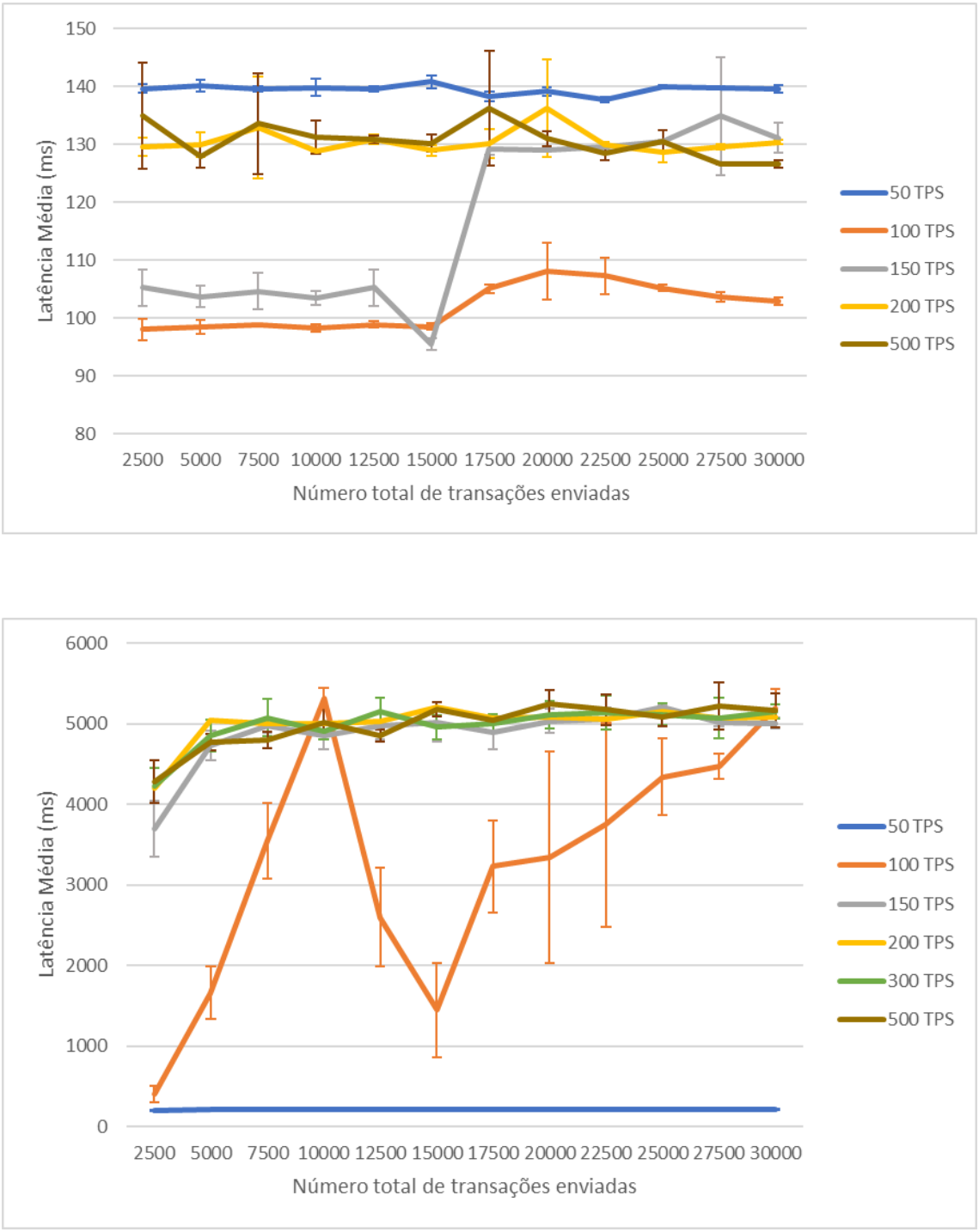


Figura 6.27: T6 - Representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Os gráficos acima correspondem à representação do comportamento da Latência Média em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* com *LevelDB* e com *CouchDB*, respetivamente. Observando os gráficos acima é possível verificar que os intervalos de confiança para os valores obtidos da *Fabric*, com *CouchDB*, são geralmente pequenos. O *send rate* de 100 TPS é o momento em que os intervalos de confiança são maiores, ou seja, há mais variação nos valores. No entanto, no caso da *Fabric*, com *LevelDB*, os intervalos de confiança já são maiores, o que indica uma grande variação nos valores recolhidos. Este aumento nos intervalos de confiança verifica-se ser maior, quanto maior é o *send rate* que está a ser estudado.

Na figura 6.28 encontra-se representado o número de transações que são bem sucedidas em função do *send rate* e do número total de transações enviadas. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

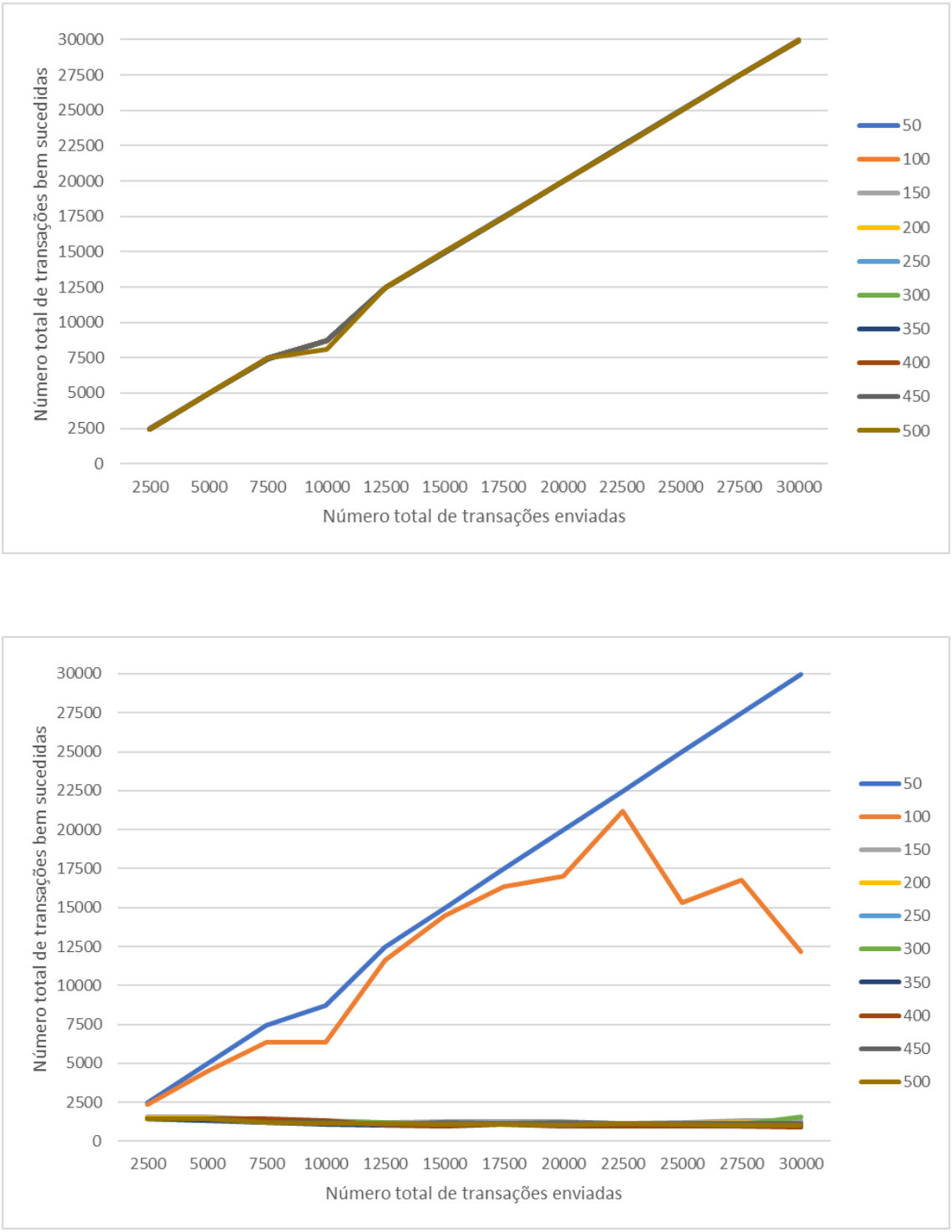


Figura 6.28: T6 - Número de transações bem sucedidas em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Observando os gráficos é possível observar que o número de transações bem sucedidas quando é utilizada a *CouchDB* é inferior à conseguida quando é usada *LevelDB*. Este facto é mais visível a partir do *send rate* de 100 TPS. Enquanto que a *Fabric* com *LevelDB* consegue tratar praticamente todos os pedidos, a *CouchDB*, a partir de um certo momento, deixa de conseguir tratar os pedidos. A partir do *send rate* de 150 TPS, que se encontra

sobreposto com todos os outros *send rates* superiores, o número de transações bem sucedidas apenas consegue alcançar perto de 2000 transações. A partir deste valor, a *CouchDB* dá a sensação de sobrecarga, não conseguindo lidar com os restantes pedidos. Este número de transações bem sucedidas pode estar na origem dos baixos valores de *throughput* apresentados de seguida, na figura 6.29.

Na figura 6.29 é apresentado o comportamento do *throughput* em função do *send rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

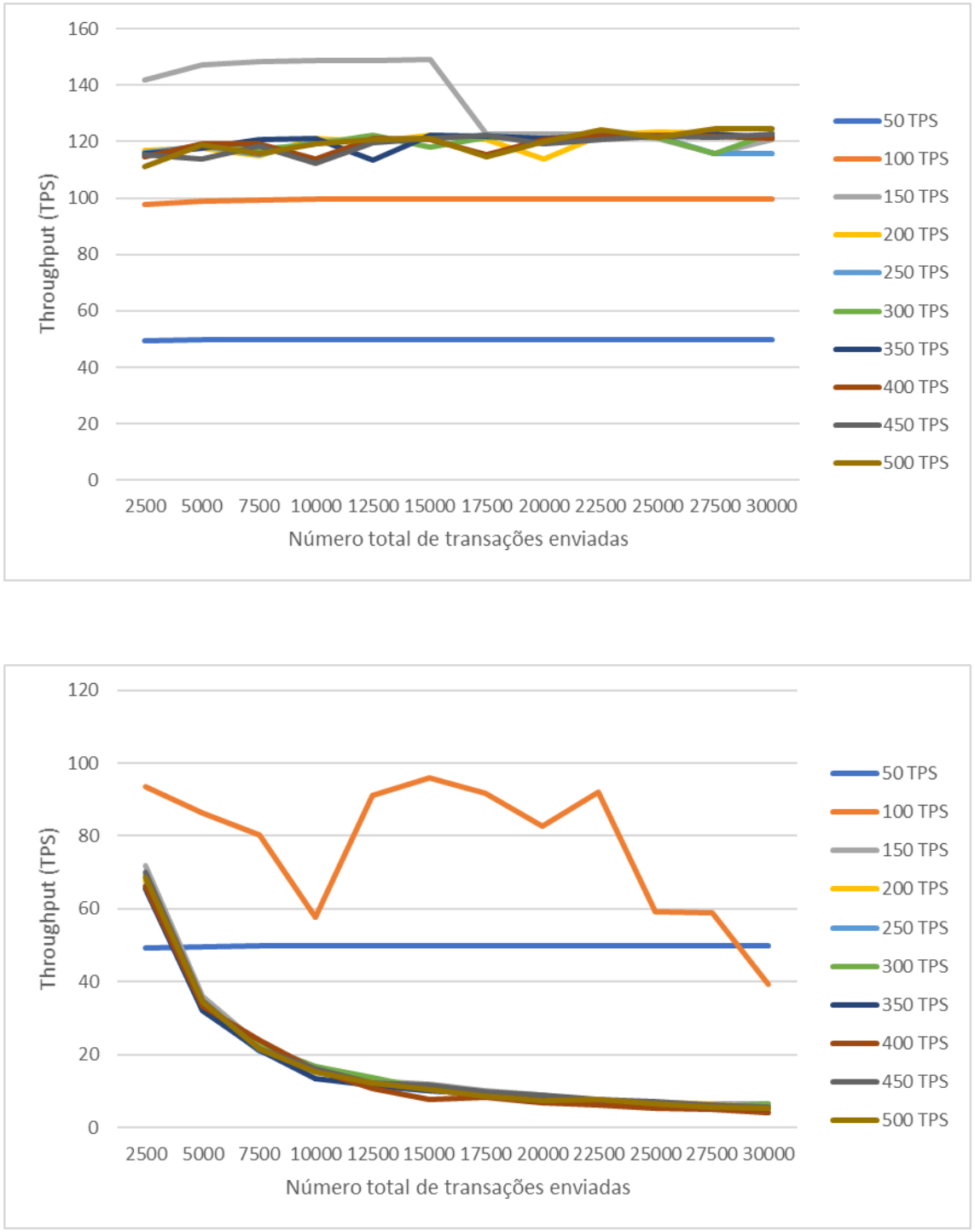
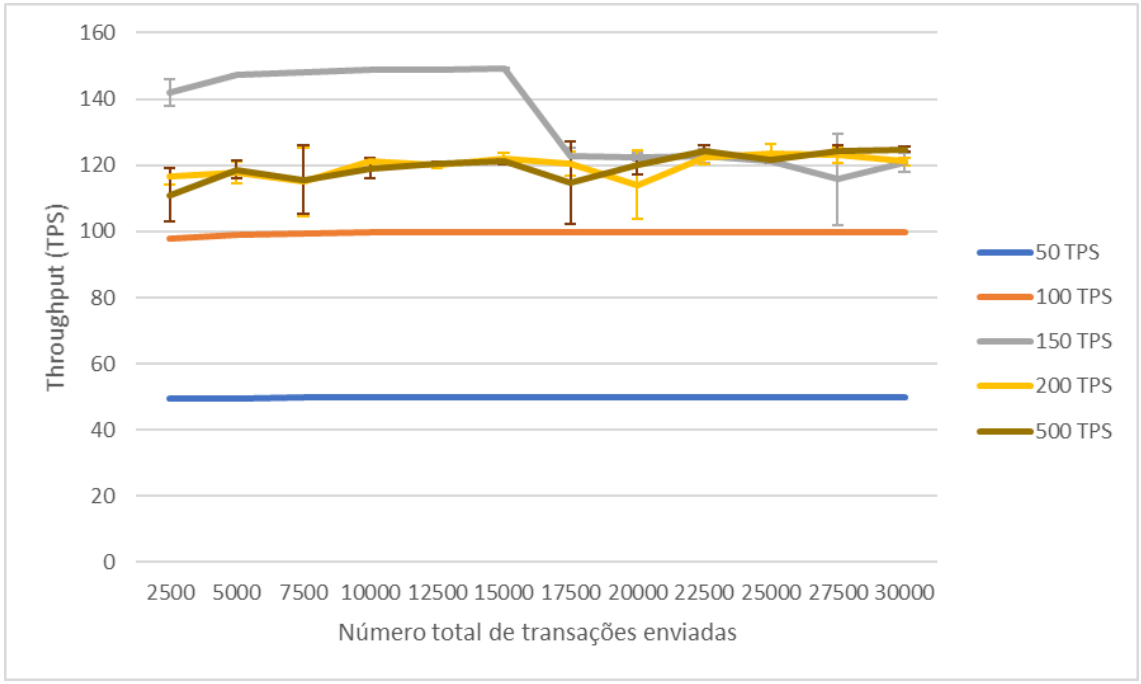


Figura 6.29: T6 - Comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Fazendo uma análise aos gráficos apresentados, o correspondente à *Fabric*, com *LevelDB*, apresenta valores constantes para *send rate* de 50 TPS e 100 TPS, existindo uma mudança nos 150 TPS. O comportamento do *throughput* torna-se mais ou menos constante por volta dos 120 TPS. Em relação à *Fabric* com *CouchDB*, o *send rates* de 50 TPS é mais ou menos constante. Existe, neste caso, uma mudança abrupta no comportamento quando o *send rate* é de 100 TPS. O *throughput* sofre muitas mudanças nos seus valores, o que pode também estar relacionado com os valores de latência apresentados anteriormente na figura 6.26. Nos restantes *send rates*, o *throughput* tende a diminuir, o que pode estar relacionado com o número de transações que são bem sucedidas, demonstrado na figura 6.28.

De seguida, na figura 6.30, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.



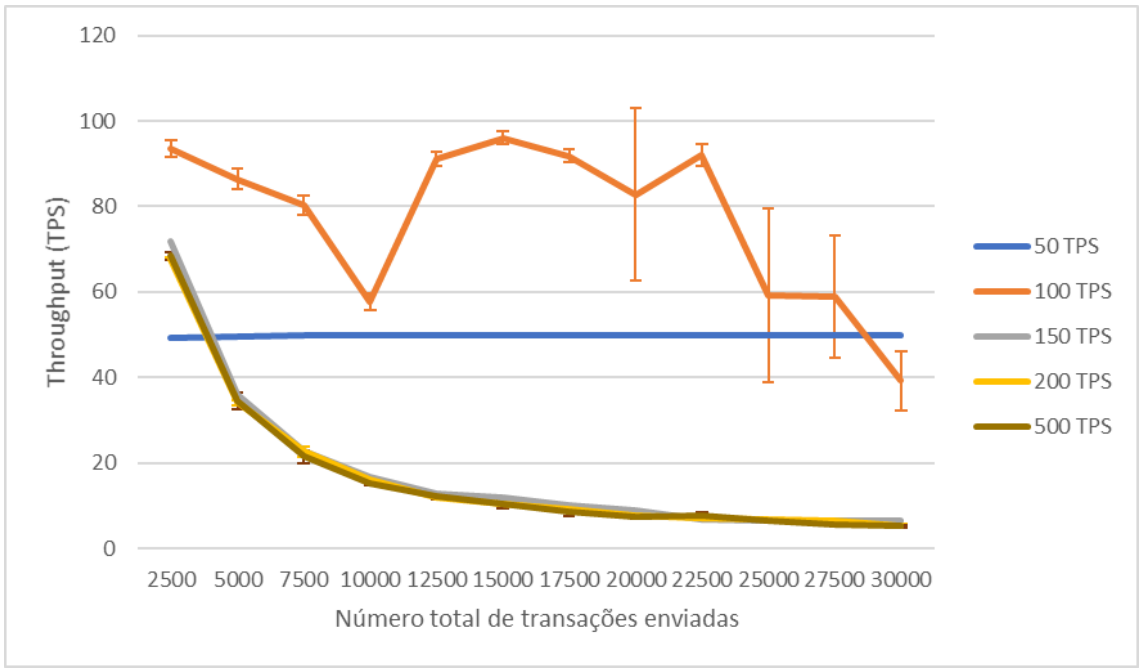


Figura 6.30: T6 - Representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Os gráficos acima correspondem à representação do comportamento do *Throughput* em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* com *LevelDB* e com *CouchDB*, respetivamente. Observando os gráficos acima é possível verificar que os intervalos de confiança para os valores obtidos da *Fabric*, em ambos os casos, são geralmente pequenos. O *send rate* de 100 TPS, a partir das 17500 transações, é o momento em que os intervalos de confiança são maiores, no caso da utilização da *CouchDB*, ou seja, há mais variação nos valores. No entanto, no caso da *Fabric*, com *LevelDB*, os intervalos de confiança são maiores quando o *send rate* é de 500 TPS.

Na figura 6.31 encontra-se representada a percentagem de CPU utilizada em função do *Send Rate* e do número total de transações que são enviadas para a *blockchain*. O primeiro gráfico diz respeito à plataforma *Fabric* com *LevelDB* e o segundo utilizando *CouchDB*.

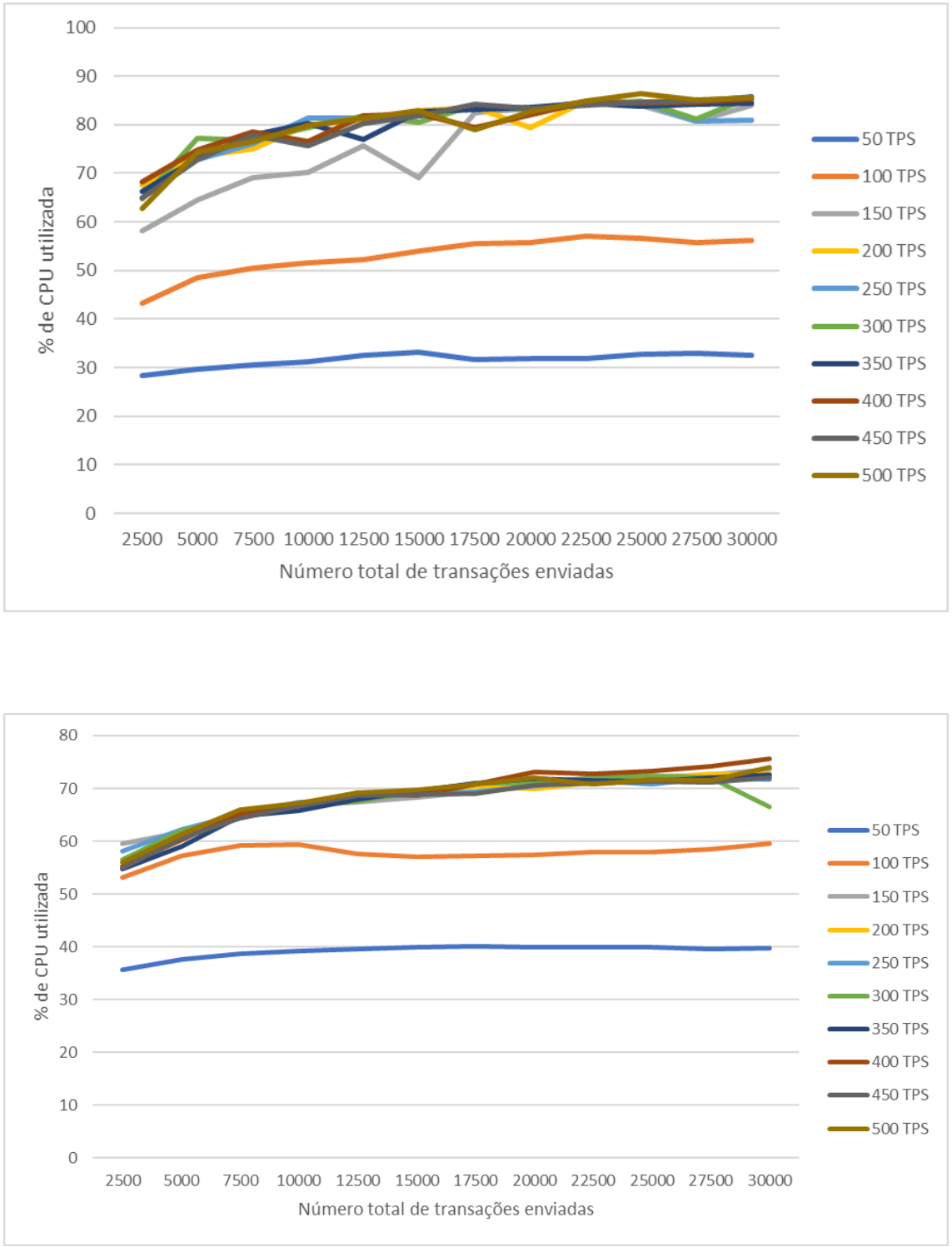


Figura 6.31: T6 - Percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Observando os gráficos é possível verificar que em ambos os casos a percentagem máxima de CPU obtida anda perto dos 80%. É ainda de referir que quando mais baixo o *send rate*, mais baixa é a percentagem de CPU que está a ser utilizada. Apenas a partir do *send rate* de 150 TPS é que os valores se vão tornando mais próximos e constantes.

De seguida, na figura 6.32, apresenta-se apenas um número reduzido dos valores apresentados anteriormente, de forma a serem visíveis os intervalos de confiança e como estes se interceptam.

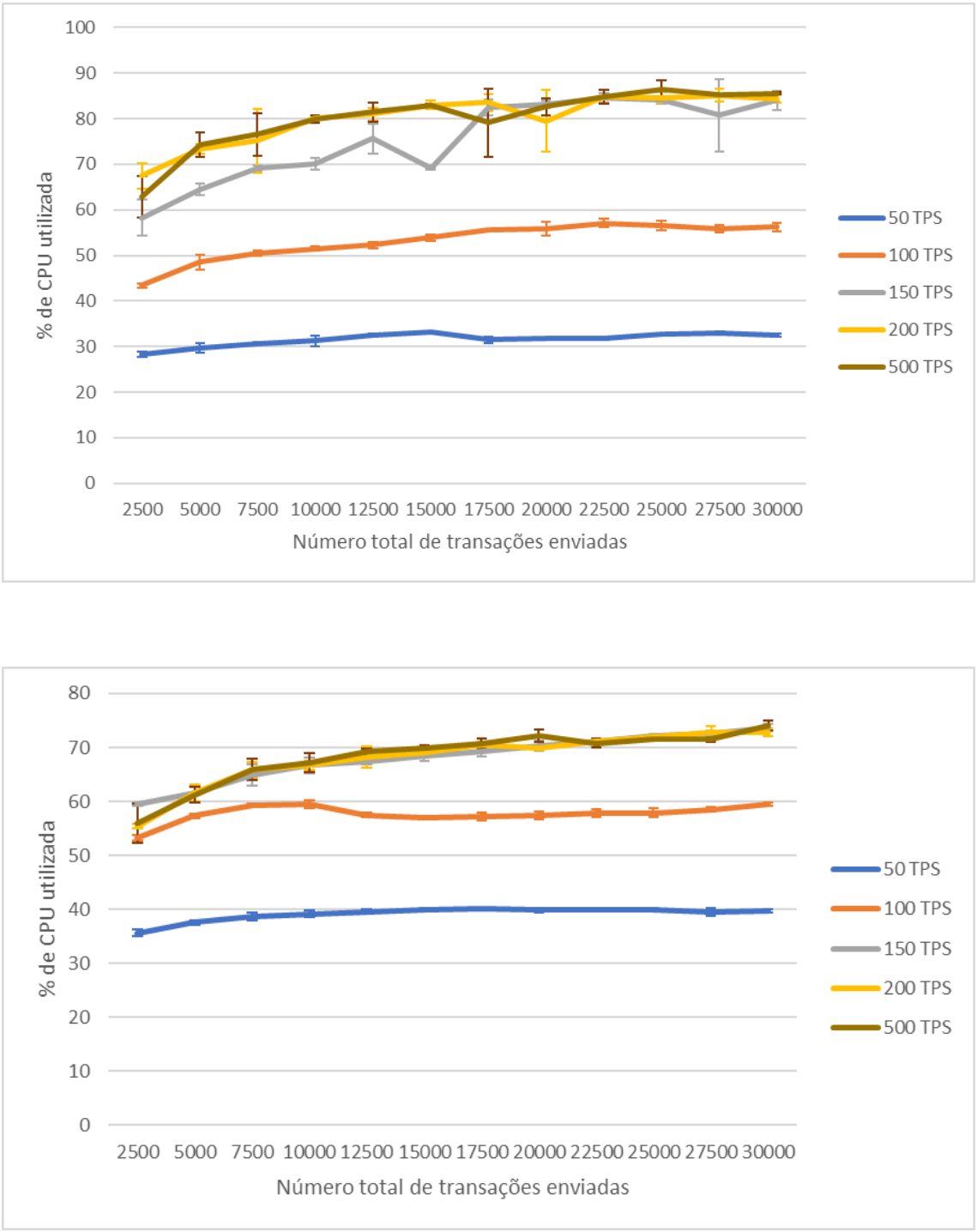


Figura 6.32: T6 - Representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com Intervalos de Confiança
Topo: Fabric - *LevelDB*, Fundo: Fabric - *CouchDB*

Os gráficos acima correspondem à representação da percentagem de CPU utilizada em função do *Send Rate* e do número total de transações enviadas com intervalos de confiança para a *Fabric* com *LevelDB* e com *CouchDB*, respetivamente. Analisando os gráficos é possível verificar que, em ambos, os intervalos de confiança para os valores obtidos são pequenos.

Por fim, o comportamento da plataforma *Hyperledger Fabric*, mais uma vez, é afectado de forma notória quando utilizada *CouchDB*. Sendo as métricas mais afectadas a latência, o *throughput* e número de transações bem sucedidas.

6.8 Conclusões gerais após a execução do *benchmark*

Da realização do *benchmarking*, através da execução do conjunto de testes nas plataformas FISCO BCOS e *Hyperledger Fabric*, é possível retirar algumas conclusões.

Em relação à comparação direta das duas plataformas, foi possível verificar que a FISCO BCOS apresenta, na grande maioria das vezes, valores de latência superiores aos obtidos pela *Fabric*. Em ambas, estes valores aumentam quando se trata de realizar operações de escrita na *blockchain*. Ou seja, o desempenho de ambas é melhor para operações de leitura sobre a *blockchain* quando comparadas com as operações de escrita, o que seria de esperar uma vez que as operações de leitura na *blockchain* são praticamente imediatas, enquanto que as operações de escrita não o são, reduzindo assim o *Throughput* e aumentando a Latência da plataforma.

Foi possível verificar ainda que a plataforma FISCO BCOS atinge valores de *Throughput* mais elevados, mas apresentando um comportamento muito variado e instável, o que vai de encontro ao que se encontra mencionado em (Liu and Huang, 2019).

Dos testes realizados relativamente à diferença entre usar ou não o mecanismo *Raft* foi possível verificar que, independentemente de utilizar ou não este mecanismo na plataforma *Fabric*, esta continua a apresentar um melhor desempenho para operações de leitura sobre a *blockchain*. De forma geral, os valores de latência são sempre mais elevados quando o *Raft* é utilizado, quer nas operações de leitura quer nas de escrita. O mesmo acontece com a percentagem de consumo de CPU. O número de transações bem sucedidas utilizando *Raft* é sempre menor que o alcançado pela *Fabric* sem *Raft*, o que seria de esperar, visto que se trata da replicação das transações por vários clientes, passando pela aprovação e validação dos mesmos.

Com a realização dos testes relativos à comparação dos resultados obtidos utilizando *LevelDB* e *CouchDB* foi possível verificar que, tal como anteriormente, independentemente de qual foi utilizada, a plataforma *Hyperledger Fabric* continua a apresentar melhor desempenho para operações de leitura sobre a *blockchain* quando comparadas com as operações de escrita. De forma geral, os valores de latência são sempre mais elevados quando é utilizada *CouchDB*, quer nas operações de leitura quer nas operações de escrita. A percentagem de consumo de CPU reage da mesma forma, sendo sempre superior quando se usa *CouchDB*. Tal como aconteceu no teste T5, os valores de *Throughput* para *CouchDB*, apesar de não ser muito notório nas operações de leitura, são um pouco mais baixos que os valores obtidos para *LevelDB*. Nas operações de escrita, esta diferença é abrupta no caso do *Throughput*. Esta pode dever-se ao facto de o número de transações bem sucedidas diminuir substancialmente quando o número de transações enviadas para a plataforma aumenta, utilizando *CouchDB*, chegando a atingir taxas de sucesso muito reduzidas. Apesar de ser possível notar todo este tipo de comportamentos quando a *Fabric* utiliza *CouchDB*,

não foi possível encontrar uma explicação para o mesmo.

Uma conclusão interessante é o facto de o comportamento da *Hyperledger Fabric* quando utiliza *LevelDB* fazendo uso do mecanismo de consenso *Raft* ser muito semelhante ao comportamento da mesma quando utiliza *CouchDB*, sem *Raft*. Mais uma vez, não foram encontradas justificações para estes resultados.

Esta página foi propositadamente deixada em branco.

Capítulo 7

Conclusão

Com a elaboração deste trabalho, ao longo do primeiro semestre, foi possível adquirir vários conhecimentos relativos à tecnologia *blockchain*. Entre eles, a definição de *blockchain*, os tipos diferentes que existem, entender em que consistem os mecanismos de consenso e os *smart contracts* e estudar o funcionamento das plataformas *blockchain* que se pretendia estudar (*Hyperledger Fabric*, *Quorum* e FISCO BCOS). Tendo sempre em consideração o foco principal do trabalho. Contudo, como explicado anteriormente, não foi possível abordar a plataforma *Quorum*, apenas tendo sido estudadas as plataformas *Hyperledger Fabric* e FISCO BCOS.

Ao longo do segundo semestre, procedeu-se à recolha e análise de trabalhos existentes na área de *benchmarking* de plataformas *blockchain*. Esta tarefa foi fundamental para o desenvolvimento deste trabalho pois deu a conhecer diferentes ferramentas de teste de *benchmark* que podiam ser utilizadas. Após este estudo, foi tomada a decisão de utilizar a ferramenta *Hyperledger Caliper*, por se tratar de um projeto recente, *OpenSource*, que permitia a recolha de diversas métricas e oferecia suporte a diversas plataformas *blockchain*. Esta ferramenta sofreu posteriormente alterações relativas à criação de um *script* que permitisse a execução dos testes relativos às plataformas *Hyperledger Fabric* e FISCO BCOS, ao desenvolvimento de outro *script* que permitiu a comparação da performance da *Fabric* quando esta utiliza ou não o mecanismo de consenso *Raft* e ainda outro *script* que permitiu verificar o funcionamento da *Fabric* utilizando *LevelDB* ou *CouchDB*. Foi também automatizado o processo de recolha dos resultados obtidos da plataforma FISCO BCOS, passando a ser possível guardar uma cópia do *benchmark* realizado. Não foi possível concluir a conexão da ferramenta à plataforma *Quorum* por não se encontrar um adaptador da mesma que fosse possível ligar à ferramenta de *benchmark* utilizada, a *Hyperledger Caliper*.

Foi então aproveitado o sistema de *benchmark* que a ferramenta disponibiliza bem como os testes que se encontram disponíveis na mesma. Com a utilização deste sistema, foi possível comparar o desempenho das duas plataformas *blockchain*, *Hyperledger Fabric* e FISCO BCOS, comparar o desempenho da *Fabric* quando faz ou não uso do mecanismo de consenso *Raft* e comparar a comportamento da mesma plataforma quando usa *LevelDB* ou *CouchDB*. Dos resultados obtidos foi possível verificar que alguns iam de encontro com os obtidos em (Dias, 2019), como por exemplo os valores de latência média e *throughput* alcançados pela *Fabric* para operações de leitura e escrita na *blockchain*. Outros resultados estavam em conformidade com conclusões apoiadas por outros autores, como é o caso de (Liu and Huang, 2019), no que diz respeito ao *throughput* alcançado pela FISCO BCOS. E outros, são apoiados pela própria arquitetura e modo de operação da plataforma

Hyperledger Fabric em si.

Para concluir, considera-se que não foram cumpridos todos os objetivos proposto, visto que inicialmente era pretendido estudar também a plataforma *Quorum*. Contudo, do trabalho realizado, foi possível utilizar as diferentes combinações que se encontram disponíveis pela ferramenta *Hyperledger Caliper*. Com a elaboração deste trabalho foram ganhos diversos conhecimentos na área da tecnologia *blockchain* que até ao momento da sua execução pouco ou nada se sabia acerca da mesma. Ao longo de todo o percurso, foram encontradas diversas dificuldades, tanto logo no começo, aquando do estudo da tecnologia, por se tratar de algo novo, como durante todo o processo. A instalação da ferramenta, a conexão das plataformas *blockchain* a esta, a execução dos testes em si, foram as principais dificuldades encontradas, que requereram algum tempo para a sua percepção.

Aquando da elaboração do trabalho presente neste documento foram identificadas algumas propostas de trabalho futuro a realizar:

- **Comparar a performance da plataforma *Hyperledger Fabric* na versão 2.0 com a versão estudada** - enquanto estava a ser elaborado o presente trabalho, foi lançada uma nova versão da plataforma *Hyperledger Fabric*, a versão 2.0. Tendo em consideração que neste documento foi estudada a versão 1.4, é feita a sugestão de comparar de que forma as modificações feitas a esta plataforma afectam o seu desempenho.
- **Avaliar a performance da plataforma *Quorum*** - foi realizada a tentativa de conectar esta plataforma à ferramenta de *benchmark Hyperledger Caliper*. Contudo, esta conexão não foi possível devido à impossibilidade de terminar esta tarefa a tempo de não influenciar e prejudicar toda a elaboração do restante trabalho. É então deixada a sugestão de voltar a tentar esta ligação.

Referências

Agrawal, H. (2017). What is Double Spending & How Does Bitcoin Handle It? https://coinsutra.com/bitcoin-double-spending/, Acedido em 2019-11-23.

Antunes, L. (2019). *Tecnologia BlockChain e Criptomoedas: O Que é Isto?* Plátano Editora.

Baliga, A., Subhod, I., Pandurang, K., and Siddhartha, C. (2018). Performance Evaluation of the Quorum Blockchain Platform. page 8.

BCOS (2020). FISCO-BCOS/FISCO-BCOS. https://github.com/FISCO-BCOS/FISCO-BCOS, Acedido em 2020-05-21.

Brakeville, S. and Perepa, B. (2019). Blockchain Basics: Introduction to Distributed Ledgers. https://developer.ibm.com/tutorials/cl-blockchain-basics-intro-bluemix-trs/, Acedido em 2019-11-23.

Buterin, V. (2015). On Public and Private Blockchains. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/, Acedido em 2019-11-25.

Caliper, H. (2019). Getting Started. https://hyperledger.github.io/caliper/vLatest/getting-started/ Acedido em 2020-01-15.

Cheah, I., Golash, R., and Ma, K. (2018). IBFT Consensus. https://whitepaper.ledgerium.io/architecture-blockchain/ibft, Acedido em 2019-12-20.

Christidis, K. and Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303. http://ieeexplore.ieee.org/document/7467408/, Acedido em 2019-12-13.

Dias, R. (2019). *Análise de Plataformas Blockchain*. PhD thesis, Universidade de Coimbra.

Dinh, T. T. A., Wang, J., Chen, G., Liu, R., Ooi, B. C., and Tan, K.-L. (2017). BLOCKBENCH: A Framework for Analyzing Private Blockchains. In *Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17*, pages 1–16, Chicago, Illinois, USA. ACM Press.

Dragonchain (2019). What Different Types of Blockchains are There? | Dragonchain - Blockchain as a Service. https://dragonchain.com/blog/differences-between-public-private-blockchains/, Acedido em 15/01/2020.

Frankenfield, J. (2019). Proof of Activity (Cryptocurrency). https://www.investopedia.com/terms/p/proof-activity-cryptocurrency.asp, Acedido em 2019-11-29.

Gronholt-Pedersen, J. (2018). Maersk, IBM to launch blockchain-based platform for global trade - Reuters.

Gupta, M. (2017). Blockchain For Dummies® IBM Limited Edition. page 51.

Heap, I. (2017). Blockchain Could Help Musicians Make Money Again. page 6.

Huang, D., Ma, X., and Zhang, S. (2018). Performance Analysis of the Raft Consensus Algorithm for Private Blockchains. *arXiv:1808.01081 [cs]*. `http://arxiv.org/abs/1808.01081`, Acedido em 2019-12-12.

Ismail, L. and Materwala, H. (2019). A Review of Blockchain Architecture and Consensus Protocols: Use Cases, Challenges, and Solutions. preprint, MATHEMATICS & COMPUTER SCIENCE. `https://www.preprints.org/manuscript/201908.0311/v1`, Acedido em 2019-12-06.

Jayachandran, P. (2017). The difference between public and private blockchain. `https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain/`, Acedido em 2019-11-25.

JPMorgan Chase & Co. (2016). Quorum. `https://docs.goquorum.com/en/latest/`, Acedido em 2019-10-18.

Klenik, A., Lincoln, N., and Ferrin, D. (2020). hyperledger/caliper-benchmarks. original-date: 2019-08-08T15:47:16Z.

Lamounier, L. (2019). Guia do Blockchain Quorum. `https://101blockchains.com/pt/blockchain-quorum/`, Acedido em 2019-12-26.

Lamport, L., Shostak, R., and Pease, M. (1982). The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):20.

Leppelsack, H. F. (2018). Experimental Performance Evaluation of Private Distributed Ledger Implementations. page 89.

Liu, J. K. and Huang, X., editors (2019). *Network and System Security: 13th International Conference, NSS 2019, Sapporo, Japan, December 15–18, 2019, Proceedings*, volume 11928 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham.

Minkenberg, M. (2015). *Transforming the Transformation?: The East European radical right in the political process*. Routledge, 1 edition.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. page 9.

Nasir, Q., Qasse, I. A., Abu Talib, M., and Nassif, A. B. (2018). Performance Analysis of Hyperledger Fabric Platforms. *Security and Communication Networks*, 2018:1–14.

Nolan, S. (2018). Quorum Blockchain Consensus Algorithms. `https://medium.com/coinmonks/quorum-blockchain-consensus-algorithms-ab38790091`, Acedido em 2019-12-17.

Ongaro, D. and Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. page 16.

PegaSys (2018). Scaling Consensus for Enterprise: Explaining the IBFT Algorithm. `https://media.consensys.net/scaling-consensus-for-enterprise-explaining-the-ibft-algorithm-ba86182ea668`, Acedido em 2019-12-16.

Pongnumkul, S., Siripanpornchana, C., and Thajchayapong, S. (2017). Performance Analysis of Private Blockchain Platforms in Varying Workloads. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. ISSN: null.

Prusty, N. (2018). *Blockchain for Enterprise*. `https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788479745/2/ch02lvl1sec21/istanbul-byzantine-fault-tolerence`, Acedido em 2019-12-20.

Rivankar, D. (2019). Gauge - Performance benchmarking tool for Hyperledger Fabric and Quorum. `https://github.com/ruipedrodias94/gauge`, Acedido em 2019-10-18.

Solanki, N. (2018). Gauge Architecture. `https://github.com/ruipedrodias94/gauge`, Acedido em 2019-10-18.

Sukhwani, H., Wang, N., Trivedi, K. S., and Rindos, A. (2018). Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network). In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–8, Cambridge, MA. IEEE.

Szabo, N. (1997). The idea of smart contracts.

Thakkar, P., Nathan, S., and Vishwanathan, B. (2018). Performance Benchmarking and Optimizing Hyperledger Fabric Blockchain Platform. *arXiv:1805.11390 [cs]*. arXiv: 1805.11390.

The Linux Foundation (2018). hyperledger-fabricdocs Documentation. page 497.

The Linux Foundation (2019). Hyperledger Caliper. Library Catalog: hyperledger.github.io.

The Linux Foundation (2019b). Ledger — hyperledger-fabricdocs master documentation. `https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html`, Acedido em 2020-05-18.

The Linux Foundation (2020). Architecture. Library Catalog: hyperledger.github.io.

WeBank, JUZIX, and Blockchain, W. (2017). FISCO BCOS Whitepaper(EN).pdf.

Yaga, D., Mell, P., Roby, N., and Scarfone, K. (2018). Blockchain technology overview. Technical Report NIST IR 8202, National Institute of Standards and Technology, Gaithersburg, MD. `https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf`, Acedido em 2019-11-23.

Zheng, Z., Xie, S., Dai, H., Chen, X., and Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In *2017 IEEE International Congress on Big Data (BigData Congress)*, pages 557–564, Honolulu, HI, USA. IEEE. `http://ieeexplore.ieee.org/document/8029379/`, Acedido em 2019-11-07.