Felipe de Souza Borsato

Avaliação das políticas best-fit e worst-fit na alocação de tarefas em grids

São José do Rio Preto

2022

Felipe de Souza Borsato

Avaliação das políticas best-fit e worst-fit na alocação de tarefas em grids

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

UNIVERSIDADE ESTADUAL PAULISTA – UNESP

CÂMPUS DE SÃO JOSÉ DO RIO PRETO

Orientador: Prof. Dr. Aleardo Manacero Junior

São José do Rio Preto

2022

B738a

Borsato, Felipe de Souza

Avaliação das políticas best-fit e worst-fit na alocação de tarefas em grids / Felipe de Souza Borsato. -- São José do Rio Preto, 2022

75 p. : il., tabs.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: Aleardo Manacero Junior

1. Grades computacionais. 2. Escalonamento. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Felipe de Souza Borsato

Avaliação das políticas best-fit e worst-fit na alocação de tarefas em grids

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Trabalho aprovado. São José do Rio Preto, 14 de janeiro de 2022:

Prof. Dr. Aleardo Manacero Junior
Orientador

Prof. Dr. Rodrigo Capobianco Guido
Professor

Profª. Drª. Adriana Barbosa Santos
Professora

São José do Rio Preto
2022

Agradecimentos

Por suas vitais contribuições ao desenvolvimento deste trabalho, registro aqui meus agradecimentos a Luis Vinicius Omar Baldissera, a meu orientador Prof. Dr. Aleardo Manacero Junior, aos integrantes da banca examinadora Prof[a]. Dr[a]. Adriana Barbosa Santos e Prof. Dr. Rodrigo Capobianco Guido e à Prof[a]. Dr[a] Renata Spolon Lobato.

Resumo

Grades computacionais são estruturas de grande utilidade para aplicações que necessitem de volumes elevados de poder computacional. Um elemento da grade que é de importância determinante a seu desempenho é o escalonador, que determinará quais recursos atenderão quais tarefas e em quais momentos. A análise de quando uma dada política de escalonamento é mais adequada a um determinado tipo de carga de trabalho pode ser feita por simulação. Neste trabalho são apresentadas simulações de diversas políticas de escalonamento, usando o iSPD (iconic Simulator of Parallel and Distributed Systems), buscando verificar como políticas orientadas pelos modelos *best-fit* e *worst-fit* se comportam em diferentes cenários de máquinas e cargas de trabalho. Os resultados obtidos permitem caracterizar melhor a adequação de cada política examinada.

Palavras-chaves: grades. escalonamento.

Abstract

Computer grids are structures very useful for applications with higher demand for computing power. The grid scheduler, which is the component in charge to atribute tasks to processors in specific moments, is a major player to determine grid's performance. The analysis about when a given scheduling policy is better suited to a given workload can be performed through simuation. In this work it is presented several simulations of different scheduling policies, using iSPD (iconic Simulator of Parallel and Distributed Systems), aiming the verification of how policies following either as best-fit or worst-fit models behave in distinct scenarios of hosts and workloads. The results achieved allow to better characterize the adequacy of each policy evaluated.

Key-words: grids. scheduling.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

O crescente poder computacional a que atualmente temos acesso permite a realização de tarefas que anteriormente poderiam ser consideradas impossíveis, especialmente no meio científico, como é o caso das simulações apresentadas por Larson (2009). Neste contexto, as grades computacionais são uma excelente fonte de processamento, por apresentarem grande escalabilidade e a possibilidade de estarem distribuídas ao redor do mundo.

A aquisição de um supercomputador é custosa e pode não ser um bom investimento. Notamos que a chamada Lei de Moore permanece aplicável (MACK, 2011), portanto podemos esperar que qualquer hardware que seja adquirido se tornará rapidamente obsoleto. Em uma arquitetura de grade, é possível combinar máquinas que diversas entidades já possuam e, posteriormente, adicionar outras de acordo com a necessidade para a expansão do poder computacional.

Para a extração eficiente do poder computacional disponível em grades, a eficiência na alocação de recursos é determinante. Neste sentido, além de apresentar boas métricas de desempenho, a política de escalonamento aplicada a uma grade computacional deve também respeitar as regras de negócio específicas da organização responsável. As possibilidades de políticas de escalonamento são diversas, com diversos objetivos, como: a aplicação do conceito de propriedade distribuída (FALAVINHA JUNIOR, 2009); o incentivo à eficiência energética (FORTE et al, 2018); a redução do impacto da transferência de dados entre diferentes nós de uma grade (CASANOVA et al, 2000); entre outros.

0.1 Motivação

A computação distribuída é um paradigma importantíssimo em diversas áreas. As contribuições de sistemas distribuídos para a ciência são muitas, dentre as quais podemos destacar projetos como o SETI@home (KORPELA, 2001), que operou de 1999 a 2020 (BANSAL, 2020) e contava com o oferecimento voluntário do poder computacional de computadores que estariam de outra forma ociosos. O Folding@home (LARSON, 2009), projeto de 2000, utiliza a mesma tecnologia para receber a colaboração de voluntários na execução de simulações de diversas proteínas, tendo recentemente possibilitado estudos a respeito do SARS-CoV-2 (CHODERA, 2020). Os dados gerados pelo *Large Hadron Collider* (LHC), maior acelerador de partículas construído, são processados por uma grade computacional (LAMANNA, 2004). Podemos notar nas últimas décadas diversas iniciativas para a construção de grades computacionais, como foi o caso da Teragrid (BECKMAN, 2005).

Os projetos citados permitem que compreendamos a relevância dos sistemas distribuídos, em específico das grades computacionais. Internamente às grades, um componente de vital importância é a política de escalonamento adotada. Para a implantação de uma grade, deve-se definir uma política que seja, além de capaz de lidar com a disponibilidade dinâmica de recursos e alocá-los eficientemente para maximizar seu desempenho, adequada à proposta da grade em questão. Como um exemplo, notamos que em um sistema como o Folding@home, em que voluntários oferecem suas máquinas para auxiliar em pesquisas, a definição da política de escalonamento não precisa levar em conta a satisfação dos usuários. Por outro lado, em grades em que diferentes administradores oferecem seus recursos com o objetivo de aumentar seu próprio poder computacional, pode ser benéfico que o escalonador busque maximizar a satisfação de todos os usuários, mesmo com a possível implicação de alguma perda de desempenho, uma vez que a satisfação dos usuários pode atrair novos colaboradores, expandindo os recursos da grade.

0.2 Objetivo

O objetivo deste trabalho é reunir informações a respeito das principais políticas de escalonamento de tarefas em grades computacionais e compará-las por meio de simulações, possibilitando melhor compreensão do comportamento de grades computacionais e suas políticas de escalonamento. Tais políticas serão aplicadas, nas simulações, em grades com diferentes disponibilidades de poder computacional e sob diferentes cargas de trabalho, o que caracteriza os diferentes cenários simulados, de forma que espera-se poder identificar quais políticas apresentam melhor desempenho de acordo com a variação destes fatores.

0.3 Organização do texto

O capítulo 2 deste texto contém fundamentação teórico a respeito de grades computacionais e escalonamento, com definições disponíveis da literatura da área. No capítulo 3 são apresentadas as políticas de escalonamento a serem estudadas neste trabalho. No capítulo 4 são definidas as métricas por meio das quais as políticas serão analisadas e os pontos que este trabalho busca esclarecer e apresentados os cenários simulados e seus resultados. Por fim, o capítulo 5 apresenta as conclusões às quais podemos chegar com os dados obtidos no capítulo 4 e possíveis direções para trabalhos futuros. As referências bibliográficas encontram-se no capítulo 5.

1 Grades computacionais

Este capítulo apresenta os conceitos de grades computacionais e escalonamento em grades por meio de definições disponíveis na literatura de sistemas distribuídos, possibilitando a compreensão das políticas a serem tratadas nos capítulos seguintes.

1.1 Definições

Coveney (2005) define grades computacionais como sistemas de computação distribuída executada de maneira transparente através de múltiplos domínios administrativos. Neste caso, o conceito de transparência é usado de maneira a indicar que um usuário da grade não precisa se preocupar com questões de grande complexidade a respeito da grade em si, podendo utilizar os recursos da grade como se o sistema que o conecta a esses recursos fosse transparente/imperceptível. Esta definição é generalista em relação aos recursos disponibilizados pela e para a grade, englobando quaisquer formas de tarefas digitais, como processamento, armazenamento de dados, entre outras.

Foster (2002) apresenta uma definição mais estrita, segundo a qual são considerados grades os sistemas computacionais que atendam aos três seguintes requisitos:

1. **Coordena recursos não sujeitos a controle centralizado**: os recursos devem estar distribuídos entre diversos domínios administrativos.
2. **Usa protocolos e interfaces abertos, generalistas e padronizados**: os protocolos utilizados devem ser abertos e padronizados, caso contrário estaríamos tratando um sistema específico.
3. **Entrega qualidades de serviço não-triviais**: o sistema deve conseguir coordenar seus recursos para oferecer qualidades de serviço que de outra forma não seriam possíveis, como tempo de resposta, disponibilidade, segurança, entre outras.

1.2 Arquitetura de uma grade

Foster (2001) divide a arquitetura de grades computacionais em cinco camadas distintas. A definição dessas camadas segue um modelo de ampulheta, em que um pequeno conjunto de abstrações e protocolos, representando a parte estreita da ampulheta, conecta um grande número de recursos a um grande número de serviços oferecidos pela grade, ambos representando as extremidades da ampulheta.

As camadas são classificadas nas seguintes definições:

1. **Estrutura**: recursos aos quais a grade promove o acesso. Tais recursos podem ser físicos, como processadores e sensores, ou lógicos, como sistemas distribuídos de arquivos e clusters.

2. **Conectividade**: protocolos de comunicação e autenticação. Os protocolos de comunicação, sendo estes tipicamente os protocolos da Internet, são responsáveis pela troca de informações entre componentes da camada de estrutura. Os protocolos de autenticação oferecem mecanismos seguros para verificar a identidade de usuários e de recursos.

3. **Recursos**: negociação e controle de recursos individuais. Construída sobre os protocolos de conectividade, esta camada trata o controle individual de recursos, ignorando o estado global da grade.

4. **Coletiva**: coordenação de múltiplos recursos. Esta camada coordena a interação entre diferentes recursos para oferecer serviços como diretórios, escalonamento, monitoramento, entre outros.

5. **Aplicações**: aplicações que serão diretamente acessadas pelos usuários. Podem ser construídas sobre qualquer uma das diferentes camadas.

1.3 Escalonador

Conforme definido por Zhu (2013), o escalonador gerencia os recursos da grade, atuando como uma interface entre os usuários e os recursos. Isto está de acordo com as camadas da arquitetura da grade definidas por Foster (2001), em que o serviço de escalonamento se encontra na camada coletiva, uma das camadas situadas na região estreita do modelo de ampulheta, ligando a camada de aplicações à camada de estrutura.

O trabalho do escalonador é alocar os adequados recursos às tarefas submetidas pelos usuários. O processo para isso se dá, conforme proposto por Schopf (2002), pelas seguintes etapas:

1. **Descoberta de recursos**: o escalonador deve, primeiramente, identificar a quais recursos o usuário que submeteu a tarefa em questão tem acesso. Dentre os elementos desse conjunto, devem ser identificados quais recursos são capazes de executar a tarefa submetida.

2. **Seleção de sistema**: identificados os recursos que podem atender a tarefa, deve ser escolhido o recurso ou conjunto de recursos que a executará. Isto se dá em dois passos: a coleta de informações sobre o estado dos recursos identificados e, então, a decisão com base nessas informações.

3. **Execução**: a etapa final é a execução da tarefa pelo recurso ou conjunto de recursos selecionado. Isso se dá com a reserva do recurso, submissão da tarefa, monitoramento do progresso, notificação do usuário e entrega dos arquivos resultantes.

1.4 Considerações finais

Foram apresentados neste capítulo os conceitos que fundamentam este trabalho, a arquitetura de uma grade computacional e o processo genérico de escalonamento em uma grade. No capítulo seguinte serão apresentadas as principais políticas de escalonamento em grades computacionais.

2 Políticas de escalonamento em grades computacionais

Neste capítulo estão compiladas as informações necessárias para a compreensão das políticas de escalonamento em grades que serão testadas. Tais políticas utilizam diversos critérios objetivando tomar as melhores decisões ao escalonar os recursos da grade.

Para os fins deste capítulo, adotaremos as seguintes definições, conforme constam em Dong (2006):

- **Tarefa**: uma unidade atômica que deve ser atribuída a algum recurso pelo escalonador.
- **Propriedades de uma tarefa**: parâmetros como requisitos de CPU ou memória.
- **Nó**: uma entidade autônoma composta por um ou mais recursos.

Em analogia aos conceitos *first-fit*, *best-fit* e *worst-fit* de alocação de memória, que podem ser vistos em maiores detalhes em Tanenbaum (2015), classificaremos as políticas aqui apresentadas conforme o seguinte:

- **first-fit**: uma tarefa é atribuída ao primeiro nó capaz de executá-la.
- **best-fit**: computado um fator de escolha, particular de cada política, a tarefa é atribuída ao nó com maior valor deste fator.
- **worst-fit**: computado um fator de escolha, particular de cada política, a tarefa é atribuída ao nó com menor valor deste fator.

2.1 Round-Robin (RR)

Round-robin é um dos mais antigos, mais simples e mais amplamente utilizados algoritmos de escalonamento em computadores pessoais. Nesta versão original desenvolvida para escalonamento em sistemas operacionais, processos alternam o uso da CPU de maneira cíclica, cada processo utilizando a CPU por um tempo determinado (quantum) até a preempção (TANENBAUM, 2015).

A aplicação a grades computacionais do round-robin foi projetada para uniformizar a quantidade de tarefas executadas pelos nós da grade. Essa política atribui tarefas aos nós da grade de maneira cíclica, de forma que um dado nó só receberá uma nova tarefa

após a atribuição de uma nova tarefa ao nó anterior. Assim, em um sistema com m nós e n tarefas, cada nó receberá n/m tarefas. A tarefa 0 será atribuída ao nó 0, a tarefa 1 será atribuída ao nó 1, ..., a tarefa m será atribuída ao nó 0, e assim por diante. Podemos ver que esse algoritmo pode causar grande ociosidade de nós mais rápidos (JAMES, 1999).

A vantagem notável do round-robin está em sua simplicidade, por não exigir a coleta de dados a respeito de propriedades dos nós da grade nem das tarefas a serem escalonadas. Classificamos este algoritmo como *first-fit*.

2.2 Workqueue (WQ) e Workqueue with Replication (WQR)

Conforme definido por Da Silva (2003), a política workqueue, assim como a round-robin, não faz uso de informações como tamanho das tarefas ou desempenho dos nós. Sempre que houver um nó disponível, é atribuída a ele uma tarefa arbitrária dentre as tarefas que aguardam escalonamento. Por, enquanto houverem tarefas, atribuir imediatamente uma nova tarefa a cada nó que se torna disponível, essa política não sofre o problema de ociosidade identificado na política round-robin. Um cenário, porém, em que a workqueue exigiria tempo excessivo para a conclusão da execução de um dado conjunto de tarefas seria quando uma das últimas tarefas escalonadas é grande e é atribuída a um nó lento. Neste caso, nós mais rápidos ficariam ociosos enquanto o nó mais lento executa a tarefa.

A fim de evitar essa situação, foi proposta por Da Silva (2003) a política workqueue with replication. Suas regras são, inicialmente, idênticas às da workqueue, se diferenciando no momento em que nós ficam disponíveis sem que hajam novas tarefas a serem executadas. Na workqueue with replication, os nós que ficariam ociosos na workqueue recebem réplicas de tarefas que ainda estão em execução por outros nós. Quando a execução de uma réplica de tarefa é encerrada, a execução das outras réplicas é interrompida.

Pode-se afirmar que as políticas WQ e WQR são de fácil implementação por conta da desnecessidade de informações a respeito do desempenho dos nós e tamanho das tarefas. Assim como o round-robin, ambas as políticas são classificadas como *first-fit* de acordo com as definições no início deste capítulo.

2.3 Dynamic Fastest Processor to Largest Task First (Dynamic FPLTF)

A política Dynamic FPLTF é uma extensão da Fastest Processor to Largest Task First, política em que as maiores tarefas são escalonadas primeiro, sendo atribuídas aos nós mais rápidos disponíveis. A maior tarefa recebe o nó mais rápido, a segunda maior tarefa recebe o segundo nó mais rápido e assim por diante, conforme definido por Menascé (1995).

Fica claro que, diferentemente das políticas anteriormente apresentadas neste trabalho, o FPLTF exige o conhecimento de propriedades das tarefas e dos nós.

A versão dynamic do FPLTF, proposta por Da Silva (2003), se diferencia da política original por utilizar informações a respeito da carga de trabalho atual de cada nó. A decisão depende das informações *velocidade do nó*, *carga do nó* e *tamanho da tarefa*. Velocidade do nó é uma medida relativa, de forma que um nó de velocidade = 2 executará uma dada tarefa na metade do tempo que seria necessário para a execução em um nó de velocidade = 1. Carga do nó indica a fração do nó que está indisponível. Por fim, tamanho da tarefa indica o tempo que seria necessário para que um nó com velocidade = 1 e carga = 0 execute a tarefa.

No início da execução do algoritmo que implementa a política Dynamic FPLTF, todos os nós recebem o valor 0 para a propriedade *tempo até estar disponível*. As tarefas são ordenadas em relação a seus tamanhos em ordem decrescente. A cada tarefa é alocado o nó que oferece o menor tempo de completude, calculado como

$$tempo\ de\ completude\ =\ tempo\ at\acute{e}\ estar\ dispon\acute{i}vel\ +\ custo\ da\ tarefa$$

em que *custo da tarefa* é:

$$custo\ da\ tarefa\ =\ \frac{\left(\frac{tempo\ da\ tarefa}{velocidade\ do\ n\acute{o}}\right)}{1\ -\ carga\ do\ n\acute{o}}$$

O valor de *tempo até estar disponível* de cada nó é acrescido do tempo relativo à tarefa ao qual o nó foi alocado. Tarefas são escalonadas até que todos os nós estejam em uso. Note que é possível que um dado nó receba mais de uma tarefa. Sempre que um nó ficar ocioso, as tarefas que não estão sendo executadas são desescalonadas e reescalonadas.

O procedimento de reescalonamento das tarefas faz com que esta política tenha boa capacidade de se adaptar ao dinamismo da grade e apresente bom desempenho, conforme constatado pelos autores. Porém, a dependência de várias informações a respeito do estado da grade faz com que esta seja uma política de difícil implementação.

Vemos que tal política se apresenta como uma política best-fit.

2.4 Sufferage

O conceito por trás da política Sufferage, proposta por Maheswaran (1999), é beneficiar as tarefas que seriam mais prejudicadas caso não fossem atribuídas ao nó que as executaria no melhor tempo. É calculado, para cada tarefa, o tempo que cada nó levaria para executá-la e, então, seu valor *sufferage*, definido como a diferença entre o menor e o segundo menor tempos. Caso o nó que oferece o melhor tempo de execução esteja

ocupado por uma tarefa que apresente *sufferage* menor, esta tarefa sofre preempção e o nó é atribuído à nova tarefa.

Nota-se que, assim como a Dynamic FPLTF, a política Sufferage precisa das informações *velocidade do nó*, *carga do nó* e *tamanho da tarefa*. Isso faz com que sua implementação seja difícil, uma vez que nem sempre essas informações estão disponíveis em um ambiente de grade computacional.

Sendo o valor *sufferage* o fator de escolha desta política, ela se caracteriza como uma política best-fit.

2.5 Min-min

A execução da política min-min ocorre em um laço de repetição em que, para cada tarefa a ser escalonada, são definidos os prazos para a execução em cada nó. Note a diferença entre *prazo para execução* e *tempo de execução* neste caso, pois *tempo de execução* denotaria apenas o tempo decorrido entre o início e o fim da execução, enquanto com *prazo para execução* tratamos do tempo total entre o momento da decisão e o momento em que a execução da tarefa se encerraria, considerando o tempo para que se encerrem outras tarefas que possivelmente estão sendo executadas pelo nó em questão. Com esses valores computados, a tarefa cuja execução pode ser encerrada mais cedo é atribuída ao nó que oferece tal prazo. A tarefa escalonada é então removida da lista de tarefas e o laço recomeça. (MAHESWARAN, 1999)

Escalonando primeiro as tarefas com melhores prazos, temos que min-min é uma política best-fit.

2.6 Max-min

A política max-min é semelhante à min-min, sendo executada por meio de um laço que, a cada iteração, ordena as tarefas em relação ao prazo em que podem ser executadas. Porém, uma vez computados tais valores, é escalonada a tarefa que apresenta maior prazo para execução. Similarmente à política min-min, após o escalonamento, a tarefa escalonada é removida da lista de tarefas e o laço recomeça.

Maheswaran (1999) constata que a política max-min pode apresentar melhor desempenho do que a min-min em casos em que exista um número muito maior de tarefas curtas do que de tarefas longas. Como um exemplo, em um caso em que existe apenas uma tarefa longa e várias tarefas curtas, vemos que, sob a política max-min, a tarefa longa seria executada concorrentemente às tarefas curtas, de forma que o tempo total para a execução de todas as tarefas seria apenas o tempo de execução da tarefa longa, enquanto,

no caso da política min-min, as tarefas curtas seriam executadas primeiro e só então a tarefa longa, o que aumentaria o tempo total de execução.

Ao contrário da política min-min, a política max-min se apresenta como uma política worst-fit ao escalonar primeiro as tarefas que oferecem os piores prazos.

2.7 Heterogeneous Owner Share Enforcement Policy (HOSEP)

A Owner Share Enforcement Policy (OSEP) (FALAVINHA JUNIOR, 2009) é uma política desenvolvida para aplicar os conceitos de propriedade distribuída e porção do proprietário em grades computacionais colaborativas, em que os usuários fornecem seus próprios recursos à grade. Essa política busca garantir que cada usuário receba, a qualquer momento em que requisitado, no mínimo uma porção de recursos equivalente àquela que forneceu. Isso parte do princípio de que usuários são estimulados a oferecem mais recursos caso tenham essa garantia (FORTE et al., 2016).

Por presumir que todo nó apresenta os mesmos recursos, a implementação original da OSEP era inadequada à maioria das grades computacionais. Assim, Forte et al. (2016) definiu uma extensão da política OSEP chamada Heterogeneous OSEP, que leva em conta as diferenças entre o poder computacional de cada nó no processo de escalonamento.

O escalonamento pela HOSEP depende das seguintes definições:

- **PowerDiff**: representa a diferença entre o poder computacional oferecido e o poder computacional recebido por um dado usuário x.

$$PowerDiff_x = \frac{Alocated_x - Provided_x}{Provided_x} \tag{2.1}$$

- **RP**: representa o poder computacional restante caso seja removido um nó y de poder computacional $Power\ y$ alocado a um usuário z.

$$RP_{z,y} = \frac{Allocated_z - Provided_z - Power_y}{Provided_z} \tag{2.2}$$

Note que, no cálculo de ambas as medidas, ocorre a divisão pelo poder computacional fornecido pelo usuário. Isso se dá para que tenhamos valores proporcionais, e não absolutos.

Considera-se que um usuário x recebeu uma porção de recursos justa quando for atendida uma das seguintes condições:

i) $PowerDiff_x \geq 0$, ou

ii) $RP_{i,j} \leq PowerDiff_x \mid \forall\ i,\ j\ alocado\ para\ i$

A ocorrência da condição i) indica que o usuário está recebendo ao menos o mesmo poder computacional que ofereceu. A ocorrência da condição ii) indica que, mesmo que o usuário x esteja recebendo menos poder computacional do que ofereceu, retirar qualquer nó presentemente utilizado por outro usuário acarretaria uma injustiça ainda maior.

Identificado um usuário que não atende a nenhum dos critérios de justiça, tal usuário deve receber a alocação de mais um nó. Será escolhido o nó cuja remoção ocasionará o maior RP, ou seja, perderá um nó o usuário que seria menos prejudicado pela preempção.

2.8 Considerações finais

Foram apresentadas e definidas as políticas a serem simuladas. Com tal conhecimento, é possível criar hipóteses a respeito do comportamento destas políticas em diferentes cenários, como será feito no capítulo seguinte. Além disso, o capítulo 4 define as métricas que serão utilizadas na avaliação da performance das políticas e apresenta o resultado dos diferentes cenários simulados.

3 Testes e resultados

Neste capítulo estão definidos todos os cenários a serem simulados. São definidos os usuários da grade, os modelos de computadores que compõem a grade, as diferentes grades simuladas, as diferentes cargas de trabalho simuladas e os tipos de tarefas que compõem tais cargas. Neste capítulo está também a formulação das hipóteses que guiarão o estudo dos resultados das simulações e uma apresentação da ferramenta por meio da qual as simulações serão executadas. Ao final do capítulo, são apresentados os resultados das simulações.

3.1 iSPD

As simulações descritas neste capítulo foram todas executadas por meio do iconic Simulator of Parallel and Distributed Systems (iSPD), apresentado por MANACERO et al. (2012), uma ferramenta que permite a definição de grades computacionais por meio de interface gráfica e a importação de políticas de escalonamento desenvolvidas por terceiros. O desenvolvimento deste trabalho exigiu a implementação das políticas Sufferage, Min-min e Max-max, enquanto as demais políticas simuladas já estavam disponíveis para o iSPD.

3.2 Métricas de desempenho

A seguir estão definidas as métricas medidas e avaliadas nos testes. Todas são calculadas automaticamente pelo iSPD durante a simulação.

- Tempo total: Tempo decorrido entre a submissão do conjunto de tarefas e o fim da execução de todas as tarefas.
- Ociosidade: Porção dos recursos que não foi utilizada, representada por uma porcentagem do total de processamento que poderia ter sido executado na grade no decorrer de todo o período entre a submissão e o fim da execução do conjunto de tarefas.
- Satisfação: Entende-se que o usuário deseja que, submetidas as tarefas, todas devem ser prontamente atendidas. Dessa forma, a métrica de satisfação do usuário para cada tarefa é calculada como a razão entre o tempo ideal e o tempo real de execução de tal tarefa. A satisfação do usuário é definida como a média de sua satisfação para cada uma de suas tarefas. Também definimos a satisfação em relação ao sistema como a média das satisfações de cada usuário.

3.3 Hipóteses

Foram formuladas as seguintes hipóteses para guiar a definição dos testes e ajudar a compreender seus resultados. Note que as hipóteses são meramente pontos a serem analisados para melhor compreensão do comportamento das grades computacionais, não necessariamente havendo a expectativa de que se mostrem verdadeiras.

1. A ociosidade de recursos e tempo total estão relacionados.
   Casos de grande ociosidade de recursos são também casos de grande tempo total.

2. Máquinas mais potentes executarão mais processamento.
   O processamento executado individualmente por máquina ao final da execução do conjunto de tarefas é proporcional ao poder computacional de tal máquina.

3. Variando-se o cenário, varia-se a política de escalonamento que apresenta as melhores métricas de desempenho.
   Não há uma política que apresente o melhor desempenho em todos os cenários simulados e em todas as métricas consideradas, sendo identificadas diferentes situações em que a adoção de diferentes políticas seja a medida ideal.

4. Os resultados são consistentes através das diferentes quantidades de poder computacional disponível.
   Dada a política que apresenta melhor desempenho em uma métrica qualquer dentre as testadas com as mesmas máquinas e carga de trabalho, tal política ainda será a de melhor desempenho em tal métrica caso o teste seja refeito com as mesmas tarefas porém com uma grade computacional diferente.

5. A distribuição do total de operações em quantidades diferentes de tarefas não afeta o tempo necessário para a completa execução do conjunto de tarefas.
   O tempo para a execução de X tarefas de tamanho Y será igual ao tempo para a execução de 2X tarefas de tamanho Y/2.

As hipóteses anteriormente apresentadas tratam do comportamento geral das políticas de escalonamento em grades. A seguir, estão formuladas também hipóteses a respeito de políticas específicas, com base nas características apresentadas no capítulo 3.

6. A política Round-Robin causa grande ociosidade dos recursos da grade.

7. Para a execução de um conjunto de tarefas com muitas tarefas curtas e poucas tarefas muito longas, a política max-min apresentará métricas de desempenho superiores às apresentadas pela política min-min.

8. A política HOSEP proporciona maior satisfação dos usuários em relação a outras políticas que não levam em consideração conceitos de justiça no escalonamento.

3.4 Ambientes simulados

Na tese em que definiu a política OSEP, Falavinha Junior (2009) descreveu testes com a política em três ambientes distintos, com 10, 20 e 40 máquinas. Com base nisso, os cenários simulados neste trabalho são em ambientes com 10, 20 e 40 máquinas, com a distinção da heterogeneidade das máquinas, uma vez que nos testes de Falavinha todas as máquinas apresentavam o mesmo poder computacional. Todas as máquinas estão ligadas diretamente a um escalonador central. Para os fins deste trabalho desconsideraremos atrasos de comunicação, portanto largura de banda e velocidade dos links não serão relevantes na especificação do sistema.

Definiremos três modelos diferentes de máquinas na grade. Esses modelos o nível de poder de processamento das máquinas, sendo que nos testes se considera, sempre que possível, uma distribuição normal para esse poder de processamento, com média 500 MFLOPS. Em particular foram definidas máquinas com o poder computacional apresentado na Tabela 1. Os valores definidos para as máquinas do tipo A e C consideram que essas máquinas totalizam cerca de 28% do total de máquinas e que seu poder computacional estaria aproximadamente a um desvio padrão do poder das máquinas do tipo B (máquinas intermediárias, com 500 MFLOPS).

Tabela 1 – Poder computacional das máquinas simuladas

Tipo de máquina	Poder computacional
Máquina A	730 MFLOPS
Máquina B	500 MFLOPS
Máquina C	270 MFLOPS

Dada a distribuição dos modelos, teremos na grade com 10 máquinas uma máquina do modelo A, oito máquinas do modelo B e uma máquina do modelo C. A grade com 20 máquinas terá duas máquinas do modelo A, 16 máquinas do modelo B e duas máquinas do modelo C. Por fim, a grade com 40 máquinas terá cinco máquinas do modelo A, 30 máquinas do modelo B e cinco máquinas do modelo C.

A fim de medir a satisfação dos usuários, definimos que as máquinas disponíveis são de propriedade de três usuários, nomeados 1, 2 e 3. O usuário 1 é o proprietário de todas as máquinas do modelo A e da mesma quantidade de máquinas do modelo B. O usuário 3 é proprietário de todas as máquinas do modelo C e da mesma quantidade de máquinas do modelo B. Por fim, o usuário 2 é proprietário das demais máquinas do modelo B. Dado o aqui definido, temos os seguintes ambientes:

Tabela 2 – Grade com 10 máquinas

	Máquina A	Máquina B	Máquina C	Processamento
Usuário 1	1	1	0	1.230 MFPLOS
Usuário 2	0	6	0	3.000 MFLOPS
Usuário 3	0	1	1	770 MFLOPS
Total	1	8	1	5.000 MFLOPS

Tabela 3 – Grade com 20 máquinas

	Máquina A	Máquina B	Máquina C	Processamento
Usuário 1	2	2	0	2.460 MFPLOS
Usuário 2	0	12	0	6.000 MFLOPS
Usuário 3	0	2	2	1.540 MFLOPS
Total	2	16	2	10.000 MFLOPS

Tabela 4 – Grade com 40 máquinas

	Máquina A	Máquina B	Máquina C	Processamento
Usuário 1	5	5	0	6.150 MFLOPS
Usuário 2	0	20	0	10.000 MFLOPS
Usuário 3	0	5	5	3.850 MFLOPS
Total	5	30	5	20.000 MFLOPS

3.5 Cargas simuladas

Falavinha (2009) definiu, em seus testes, como tarefas de curta duração aquelas executadas em até 600 segundos, tarefas de média duração aquelas executadas entre 600 a 3600 segundos e tarefas de longa duração aquelas que precisem de mais de 3600 segundos para serem executadas. Seguindo tais definições, neste trabalho os conjuntos de tarefas simulados são compostos por tarefas curtas, cuja execução leva 600 segundos em uma máquina do modelo B, médias, cuja execução leva 3600 segundos em uma máquina do modelo B, e longas, cuja execução leva 7200 segundos em uma máquina do modelo B.

3.6 Simulações

Abaixo estão descritos os cenários simulados e apresentadas as métricas obtidas em cada simulação.

3.6.1 18 tarefas de todos os tamanhos distribuídas uniformemente

A carga deste teste é de seis tarefas curtas, seis tarefas médias e seis tarefas longas. Cada usuário foi responsável pela submissão de duas tarefas de cada tamanho. Podemos considerar que esta é uma carga de relativamente poucas tarefas, visto que na grade de 20 máquinas existem máquinas suficientes para que todas as tarefas sejam imediatamente atendidas, mesmo sem utilizar as máquinas menos potentes da grade (modelo C).

- 10 máquinas:

Tabela 5 – Métricas para 18 tarefas de todos os tamanhos distribuídas uniformemente na grade de 10 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	14444,4s	49.6166%	80,84%
Workqueue with replication	12739.7s	0%	80%
Dynamic FPLTF	12000.0s	46.3549%	83,71%
Sufferage	15000.0s	55.3569%	87,1%
Min-min	12000.0s	46.5124%	88,76%
Max-min	11400.0s	39.2998%	81,4%
HOSEP	14444.4s	49.6166%	83%

  - Política com menor tempo total: Min-min (11.400 segundos)
  - Política com maior satisfação média: Min-min (88,76%)

- 20 máquinas:

Tabela 6 – Métricas para 18 tarefas de todos os tamanhos distribuídas uniformemente na grade de 20 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	13333.3s	71.0418%	99,99%
Workqueue with replication	14444.4s	0%	87,64%
Dynamic FPLTF	7200.00s	55.6507%	105,11%
Sufferage	7200.00s	55.7820%	107%
Min-min	7200.00s	55.7820%	107%
Max-min	7200.00s	55.6507%	105,1%
HOSEP	7200.00s	54.2066%	105,11%

  - Políticas com menor tempo total: Dynamic FPLTF, Sufferage, Min-min, Max-min e HOSEP (7.200 segundos)
  - Política com maior satisfação média: Min-min (107%)

- 40 máquinas:

  - Políticas com menor tempo total: Dynamic FPLTF, Sufferage, Min-min, Max-min e HOSEP (7.200 segundos)
  - Políticas com maior satisfação média: Dynamic FPLTF, Sufferage, Min-min e Max-min (115,56%)

Tabela 7 – Métricas para 18 tarefas de todos os tamanhos distribuídas uniformemente na grade de 40 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	13333.3s	84.1858%	99,99%
Workqueue with replication	12131.5s	0%	94,85%
Dynamic FPLTF	7200.00s	79.5320%	115,56%
Sufferage	7200.00s	79.5320%	115,56%
Min-min	7200.00s	79.5320%	115,56%
Max-min	7200.00s	79.5320%	115,56%
HOSEP	7200.00s	78.7443%	112,77%

3.6.2 180 tarefas de todos os tamanhos distribuídas uniformemente

A carga deste teste é de 60 tarefas curtas, 60 tarefas médias e 60 tarefas longas. Cada usuário foi responsável pela submissão de 20 tarefas de cada tamanho. Podemos considerar que esta é uma carga de relativamente muitas tarefas, uma vez que o número de tarefas é 4,5 vezes maior que o número de máquinas disponíveis na maior grade a ser testada.

- 10 máquinas:

Tabela 8 – Métricas para 180 tarefas de todos os tamanhos distribuídas uniformemente na grade de 10 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	137778s	47.3876%	19,22%
Workqueue with replication	72000.0s	5.02042%	19,18%
Dynamic FPLTF	71917.8s	5.29837%	19,3%
Sufferage	72600.0s	6.68105%	19,43%
Min-min	75000.0s	9.54123%	20,1%
Max-min	72328.8s	5.93325%	19,1%
HOSEP	72000.0s	4.88914%	20,95%

  - Política com menor tempo total: Dynamic FPLTF (71.917,8 segundos)
  - Política com maior satisfação média: HOSEP (20,95)

- 20 máquinas:

  - Política com menor tempo total: Workqueue with replication (28.800 segundos)
  - Política com maior satisfação média: HOSEP (36,84%)

- 40 máquinas:

  - Política com menor tempo total: HOSEP (22.633,2 segundos)
  - Política com maior satisfação média: HOSEP (51,07%)

Tabela 9 – Métricas para 180 tarefas de todos os tamanhos distribuídas uniformemente na grade de 20 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	68888.9s	47.2712%	32,31%
Workqueue with replication	28800.0s	0%	31,48%
Dynamic FPLTF	45000.0s	24.8526%	31,41%
Sufferage	40684.9s	14.2398%	32,57%
Min-min	40800.0s	16.5428%	33,36%
Max-min	40684.9s	17.0872%	31,38%
HOSEP	37800.0s	9.58892%	36,84%

Tabela 10 – Métricas para 180 tarefas de todos os tamanhos distribuídas uniformemente na grade de 40 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	46666.7s	60.3096%	44,77%
Workqueue with replication	36000.0s	56.2130%	42,45%
Dynamic FPLTF	25068.5s	33.1094%	46,74%
Sufferage	26926.0s	38.0457%	46,97%
Min-min	27534.2s	39.9393%	46,41%
Max-min	28888.9s	40.9155%	46,06%
HOSEP	22633.2s	23.8802%	51,07%

3.6.3 Poucas tarefas grandes e muitas tarefas pequenas

A carga utilizada para este teste é de duas tarefas grandes e 24 tarefas pequenas. Estão atribuídas ao usuário 1 uma tarefa grande e quatro pequenas, ao usuário 2 outra tarefa grande e quatro pequenas e ao usuário 3 todas as demais tarefas pequenas (16), de forma que todos os usuários estão solicitando quantidades iguais de processamento.

Note que o tempo necessário para a execução de 12 tarefas pequenas é equivalente ao tempo necessário para a execução de uma tarefa grande. Uma vez que o objetivo desta simulação é verificar a validade da hipótese 7, esta carga foi simulada apenas no ambiente com 10 máquinas, tendo em vista que com muitas máquinas disponíveis todas as tarefas seriam executadas simultaneamente e não obteríamos qualquer dado relevante.

Tabela 11 – Métricas para poucas tarefas grandes e muitas tarefas pequenas na grade de 10 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	15000.0s	80.0298%	66,98%
Workqueue with replication	8810.96s	4.08582%	78,53%
Dynamic FPLTF	8400.00s	67.8064%	70,88%
Sufferage	8400.00s	67.8064%	70,11%
Min-min	7800.00s	65.5723%	74,09%
Max-min	8400.00s	67.8064%	70,11%
HOSEP	7800.00s	62.2090%	70,9%

- Políticas com menor tempo total: Min-min e HOSEP (7.800,00 segundos)
- Política com maior satisfação média: Workqueue with replication (78,53%)

3.6.4 30 tarefas grandes

Nesta simulação cada usuário submete 10 tarefas grandes.

Tabela 12 – Métricas para 30 tarefas grandes na grade de 10 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	40000.0s	43.1014%	61,11%
Workqueue with replication	14794.5s	0%	89,21%
Dynamic FPLTF	24657.5s	14.5126%	61,79%
Sufferage	24657.5s	14.5126%	61,79%
Min-min	24657.5s	14.5126%	61,79%
Max-min	24657.5s	14.5126%	61,79%
HOSEP	31598.2s	27.5098%	61,79%

- Política com menor tempo total: Workqueue with replication (14.794,5 segundos)
- Política com maior satisfação média: Workqueue with replication (89,21%)

3.6.5 360 tarefas curtas

Nesta simulação cada usuário submete 120 tarefas curtas. Note que trata-se da mesma quantidade de processamento necessária na simulação com 30 tarefas grandes, porém tal quantidade está dividida em mais tarefas.

Tabela 13 – Métricas para 360 tarefas curtas na grade de 10 máquinas

	Tempo total	Ociosidade	Satisfação média
Round-Robin	40000.0s	43.1014%	11,59%
Workqueue with replication	22222.2s	2.62356%	11,65%
Dynamic FPLTF	22191.8s	3.12099%	11,65%
Sufferage	22191.8s	3.12099%	11,65%
Min-min	22191.8s	3.12099%	11,65%
Max-min	22191.8s	3.12099%	11,65%
HOSEP	22011.0s	1.44691%	11,16%

- Políticas com menor tempo total: Dynamic FPLTF, Sufferage, Min-Min e Max-min (22.191,8 segundos)
- Políticas com maior satisfação média: Workqueue with replication, Dynamic FPLTF, Sufferage, Min-Min e Max-min (11,65%)

3.7 Verificação das hipóteses

Com base nos resultados obtidos com as simulações, verificaremos nessa seção as hipóteses definidas na seção 3.3.

3.7.1 A ociosidade de recursos e tempo total

A respeito desta hipótese, é interessante verificar os dois casos extremos: Round-Robin, que é a política que causa maior ociosidade de recursos, e Workqueue with replication, que é a política que causa menor ociosidade de recursos. No caso da Round-Robin, notamos sempre grandes medidas de tempo total. A Workqueue with replication, porém, apesar de sempre provocar menor ociosidade de recursos, não é consistentemente a política com melhor tempo total. Isto ocorre porque grande parte do processamento realizado sob a política WQR é perdido em preempções.

3.7.2 Processamento e poder computacional

Apesar de se mostrar verdadeira em vários cenários, a hipótese de que máquinas de maior poder computacional realizarão mais processamento não pode ser tomada como regra. Em vários casos as máquinas do modelo A, mais potentes, executaram menos processamento do que até mesmo máquinas do modelo C, menos potentes. Notamos que tal hipótese é sempre válida em grades que adotem o Workqueue with Replication, que faz com que todos os nós estejam constantemente executando tarefas, de forma que o processamento total realizado por cada nó será sempre proporcional ao seu poder computacional, neste trabalho medido em MFLOPS.

3.7.3 Adequação da política de escalonamento a cada cenário

Nenhuma política testada apresentou os melhores resultados em alguma métrica em todos os cenários simulados. Com isso, observamos que de fato a variação dos cenários provoca variação na política de escalonamento mais adequada, inclusive considerando que se busque maximizar, uma vez que a política que execute as tarefas em menor tempo total não é necessariamente a politica que gere maior satifação dos usuários.

Dito isto, podemos notar também que, apesar de não haver uma política ideal a todos os casos, podemos identificar as políticas que mais frequentemente apresentam as melhores métricas. A política Min-min foi a que executou as tarefas em menor tempo total mais vezes nas simulações descritas neste trabalho. É possível também identificar as políticas que apresentam as piores métricas: em nenhum dos cenários simulados a política Round-Robin possibilitou a execução das tarefas em menor tempo total ou a maior satisfação dos usuários.

3.7.4 Variação do poder computacional disponível

Apesar de não haver garantia, parece ser relativamente seguro esperar que a política de melhor desempenho para dadas carga e grade será a política de melhor desempenho para a mesma carga em uma grade diferente. Verifiquemos os resultados dos testes com variação da grade, os testes com 18 e 180 tarefas. Sob a carga de 18 tarefas, a política Min-min foi consistentemente a melhor tanto em tempo total quanto em satisfação média. Houve variação, no entanto, no sentido de que, conforme aumentada a disponibilidade de máquinas na grade, outras políticas atingiram o desempenho da Min-min. Sob a carga de 180 tarefas, a política HOSEP foi consistentemente a política a apresentar maior satisfação média. O menor tempo total, porém, variou entre Dynamic FPLTF, Workqueue with replication e HOSEP.

3.7.5 Tempo de execução e carga das tarefas

Quando substituída por uma quantia equivalente de tarefas menores, a carga de trabalho apresentou métricas notavelmente piores, com os melhores tempo total e satisfação média, 14.794,5 e 89,21% respectivamente, se tornando 22.191,8 e 11,65%. Assim, verificamos que a distribuição da carga de trabalho em diferentes quantidades de tarefas afeta o desempenho, negando a hipótese. Os resultados indicam que o agrupamento em tarefas maiores pode beneficiar o desempenho.

3.7.6 Ociosidade de recursos e a política Round-Robin

Conforme o esperado dada a definição da política, nas simulações realizadas verificamos que a Round-Robin é a política que gera a maior ociosidade de recursos. Tal ociosidade faz com que esta seja consistentemente a política com piores métricas de desempenho, tanto tempo total quanto satisfação dos usuários. Apesar disso, é importante ressaltar que, conforme discutido na hipótese 1, menor ociosidade de recursos não necessariamente implica em menor tempo total.

3.7.7 Desempenho em cenários com muitas tarefas curtas

Esperava-se que, em um cenário com muitas tarefas curtas e poucas tarefas longas, a política Max-min proporcionasse desempenho melhor que a política Min-min. A simulação de um cenário como esse nos mostrou o resultado contrário, tendo a Min-min executado as tarefas em tempo total ligeiramente menor e resultando em satisfação média ligeiramente maior. Os dados apresentados nesse trabalho não permitem que estudemos mais profundamente os mecanismos que levaram a esse resultado.

3.7.8 Justiça no escalonamento e satisfação do usuário

A política HOSEP foi a política de maior satisfação média em um terço dos cenários simulados. Entendemos que nos cenários simulados a HOSEP não apresentou majoritariamente as melhores médias de satisfação pois, em todos os testes, todas as tarefas de todos os usuários foram submetidas simultaneamente, o que fez com que todos os usuários tivessem as mesmas oportunidades de receber recursos mesmo em algoritmos que não consideram conceitos de justiça. Em cenários em que a grade está constantemente operando, a HOSEP geraria maior satisfação por garantir que uma dada porção dos recursos será prontamente alocada a qualquer usuário assim que este submeter tarefas, o que não seria o caso sob as demais políticas aqui descritas de testas.

3.8 Considerações finais

Neste capítulo foram definidos detalhadamente os cenários simulados e os resultados das simulações. Tais resultados foram analisados para obtermos conclusões a respeito das hipóteses aqui formuladas.

4 Conclusão

As simulações apresentadas nesse texto nos permitem observar o comportamento das grades computacionais conforme as variações de poder computacional e carga de trabalho. A compreensão dos resultados é possibitada pela revisão bibliográfica apresentada nos capítulos iniciais e pelas hipóteses verificadas no capítulo 3.

4.1 Considerações Finais

Podemos concluir que, mesmo em ambientes com todas as variáveis controladas, o comportamento de uma grade computacional é difícil de prever. Em situações reais, com todo o dinamismo do sistema, torna-se uma tarefa impossível determinar uma política de escalonamento absoluta. Apesar disso, os testes promovidos neste trabalho ajudam a compreender melhor o comportamento das grades e tomar boas decisões na gestão de uma grade.

Em um cenário em que a obtenção de dados sobre o status das máquinas é muito difícil, uma grade ainda pode operar de maneira satisfatória. A política Workqueue with replication, que é de simples implementação e exige pouca informação sobre a grade, apresentou resultados equiparáveis aos de políticas mais complexas. Em vista de casos como aquele em que esperava-se que a Max-min apresentasse resultados melhores do que a Min-min, ressalta-se a importância de testar as políticas e também a importância de ferramentas de simulação para tais testes.

A política Min-min foi a política que em mais cenários executou as tarefas no menor tempo, de forma que, se a coleta das informações necessárias para a execução de tal política for possível, ela pode ser uma boa escolha generalista para obtenção de bom desempenho.

A política HOSEP foi, empatada com a Dynamic FPLTF, a segunda política a gerar os melhores resultados de tempo de execução nos cenários testados. Sendo suas métricas de performance muito próximas das métricas da política mais rápida, Min-min, a HOSEP pode ser uma boa escolha por incentivar que os participantes da grade ofereçam mais poder computacional.

4.2 Trabalhos futuros

Os resultados deste trabalho podem indicar também direções para futuras pesquisas. Destacamos as seguintes:

4.2.1 Agrupamento da carga de trabalho em diferentes quantidades de tarefas

Notamos que, apesar de apresentarem um mesmo total de operações a serem realizadas, cargas de trabalho agrupadas em diferentes quantidades de tarefas criam resultados discrepantes. O estudo de tal comportamento pode ser relevante para as grades computacionais.

4.2.2 Uso de replicação de tarefas

A Workqueue with replication, apesar de realizar o escalonamento de maneira simplista, obteve bons resultados em várias simulações por conta do uso de replicações. A aplicação da replicação de tarefas em outras políticas pode gerar bons resultados.

Referências

BANSAL, Rajeev. SETI 2020: The Pause That Refreshes?[Turnstile]. **IEEE Antennas and Propagation Magazine**, v. 62, n. 4, p. 116-116, 2020.

BECKMAN, Peter H. Building the teragrid. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, v. 363, n. 1833, p. 1715-1728, 2005.

CASANOVA, Henri et al. Heuristics for scheduling parameter sweep applications in grid environments. In: **Proceedings 9th Heterogeneous Computing Workshop (HCW 2000)(Cat. No. PR00556)**. IEEE, 2000. p. 349-363.

CHODERA, John. FOLDING@HOME UPDATE ON SARS-COV-2. **Folding@home**, 2020. Disponível em <https://foldingathome.org/2020/03/10/covid19-update/> Acesso em <1 jun. 2020>

COVENEY, Peter V. Scientific grid computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, v. 363, n. 1833, p. 1707-1713, 2005.

DA SILVA, Daniel Paranhos; CIRNE, Walfredo; BRASILEIRO, Francisco Vilar. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: **European Conference on Parallel Processing**. Springer, Berlin, Heidelberg, 2003. p. 169-180.

DONG, Fangpeng; AKL, Selim G. Scheduling algorithms for grid computing: State of the art and open problems. **School of Computing, Queen's University, Kingston, Ontario**, p. 1-55, 2006.

FALAVINHA JUNIOR, José Nelson. Escalonamento de tarefas em sistemas distribuídos baseado no conceito de propriedade distribuída. 2009.

FORTE, C. H. V. et al. Política de escalonamento Owner-Share para sistemas de grades heterogêneas. In: **Anais do XVII Simpósio em Sistemas Computacionais de Alto Desempenho**. SBC, 2016. p. 227-238.

FORTE, Cássio HV et al. An Energy-aware Task Scheduler Based in Ownership Fairness Applied to Federated Grids. In: **2018 IEEE Symposium on Computers and Communications (ISCC)**. Ieee, 2018. p. 00030-00033.

FOSTER, Ian. What is the grid? a three point checklist. **GRID today**, v. 1, n. 6, p. 32-36, 2002.

FOSTER, Ian; KESSELMAN, Carl; TUECKE, Steven. The anatomy of the grid: Enabling scalable virtual organizations. **The International Journal of High Performance Computing Applications**, v. 15, n. 3, p. 200-222, 2001.

JAMES, Heath A. et al. Scheduling independent tasks on metacomputing systems. In: **Proceedings of Parallel and Distributed Computing Systems (PDCS'99), Fort Lauderdale, Florida**. 1999.

KAY, Judy; LAUDER, Piers. A fair share scheduler. **Communications of the ACM**, v. 31, n. 1, p. 44-55, 1988.

KORPELA, Eric et al. SETI@ home-massively distributed computing for SETI. **Computing in science & engineering**, v. 3, n. 1, p. 78-83, 2001.

LAMANNA, Massimo. The LHC computing grid project at CERN. **Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment**, v. 534, n. 1-2, p. 1-6, 2004.

LARSON, Stefan M. et al. Folding@ Home and Genome@ Home: Using distributed computing to tackle previously intractable problems in computational biology. arXiv preprint arXiv:0901.0866, 2009.

MACK, Chris A. Fifty years of Moore's law. **IEEE Transactions on semiconductor manufacturing**, v. 24, n. 2, p. 202-207, 2011.

MAHESWARAN, Muthucumaru et al. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. **Journal of parallel and distributed computing**, v. 59, n. 2, p. 107-131, 1999.

MANACERO, Aleardo et al. iSPD: an iconic-based modeling simulator for distributed grids. In:**SpringSim (ANSS)** 2012. p. 5.

MENASCÉ, Daniel A. et al. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. **Journal of Parallel and Distributed Computing**, v. 28, n. 1, p. 1-18, 1995.

SCHOPF, Jennifer M. A general architecture for scheduling on the grid. **Special issue of JPDC on Grid Computing**, v. 4, 2002.

TANENBAUM, Andrew S.; BOS, Herbert. **Modern operating systems**. Pearson, 2015.

ZHU, Yanmin; NI, Lionel M. A Survey on Grid Scheduling Systems. **Technical Report SJTU_CS_TR_200309001, Department of Computer Science and Engineering, Shanghai Jiao Tong University**, 2013.

APÊNDICE A – 18 tarefas uniformes - Processamento em cada nó

Neste apêndice é apresentado, na forma de gráficos, o processamento realizado por cada máquina na simulação de 18 tarefas uniformes.

A.1 10 máquinas

Seguem os gráficos de tempo de ocupação gerados para testes com 10 máquinas.

Figura 1 – Processamento realizado por cada máquina com Round-Robin

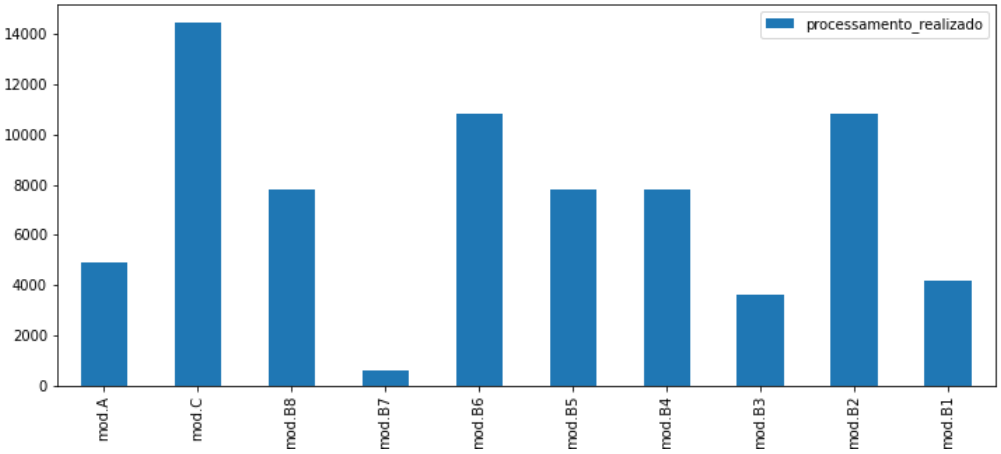


Figura 2 – Processamento realizado por cada uma das 10 máquinas com Workqueue with replication

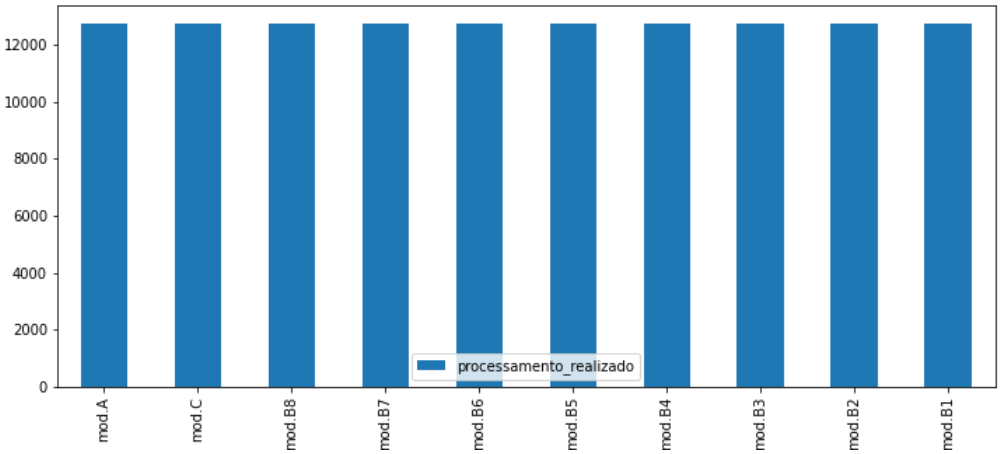


Figura 3 – Processamento realizado por cada uma das 10 máquinas com Dynamic FPLTF

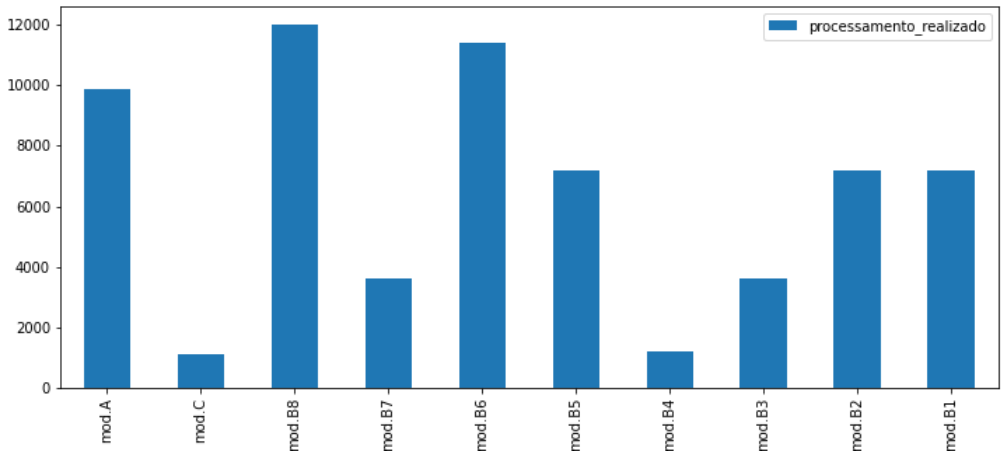


Figura 4 – Processamento realizado por cada uma das 10 máquinas com Sufferage

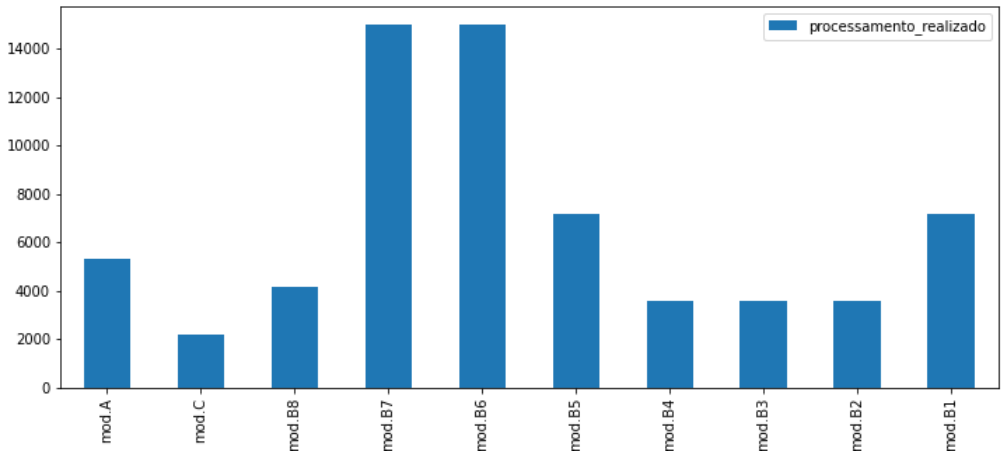


Figura 5 – Processamento realizado por cada uma das 10 máquinas com Min-min

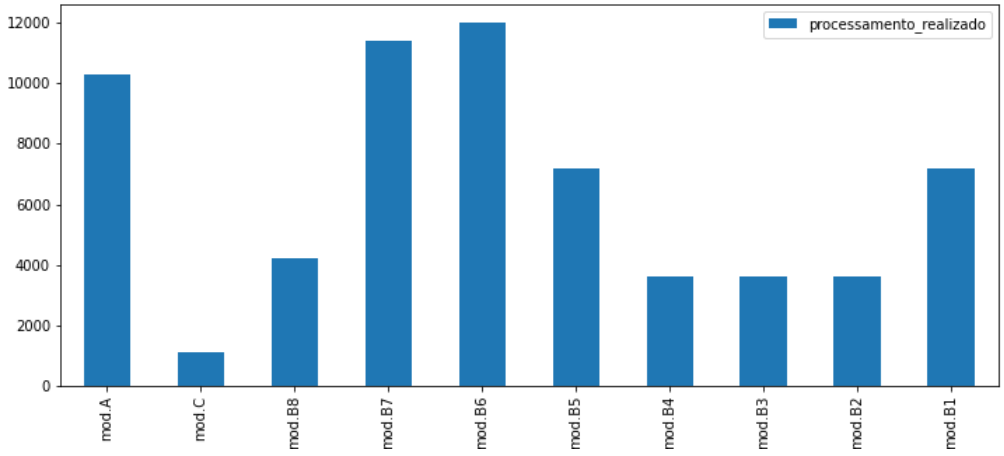


Figura 6 – Processamento realizado por cada uma das 10 máquinas com Max-min

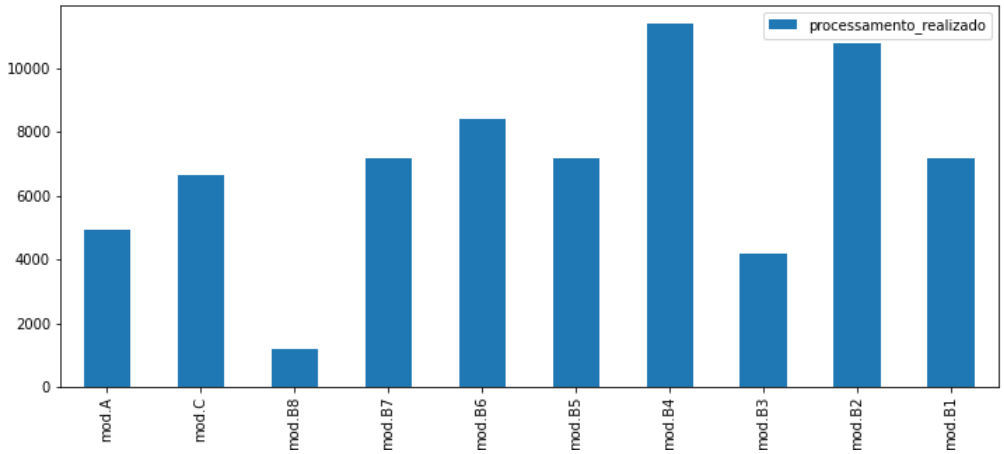
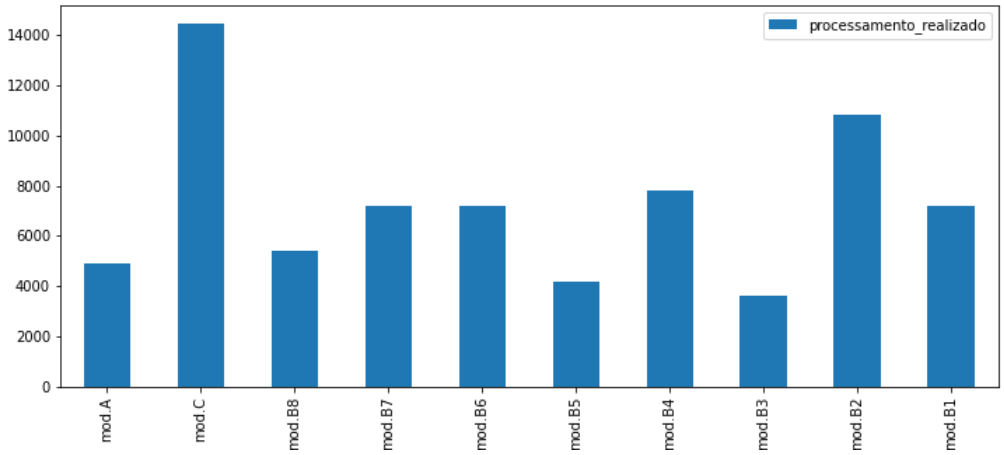


Figura 7 – Processamento realizado por cada uma das 10 máquinas com HOSEP



A.2 20 máquinas

Seguem os gráficos de tempo de ocupação gerados para testes com 20 máquinas.

Figura 8 – Processamento realizado por cada uma das 20 máquinas com Round-Robin

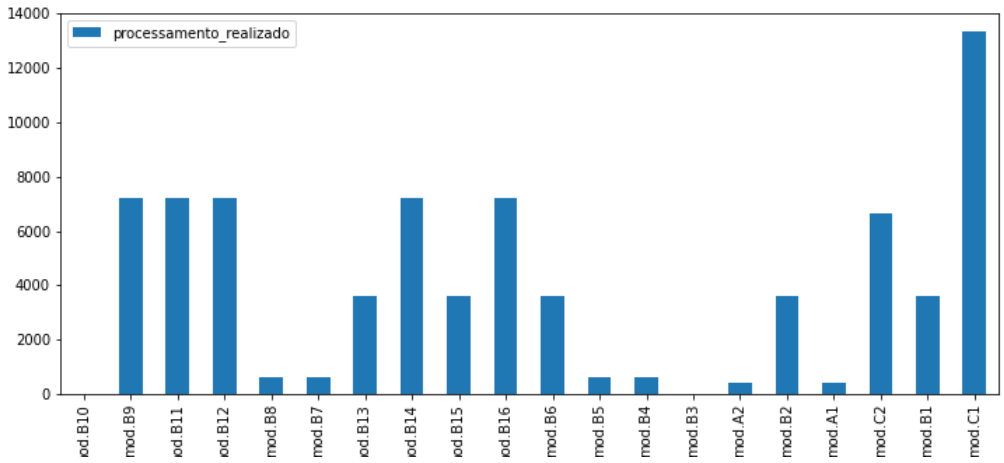


Figura 9 – Processamento realizado por cada uma das 20 máquinas com Workqueue with replication

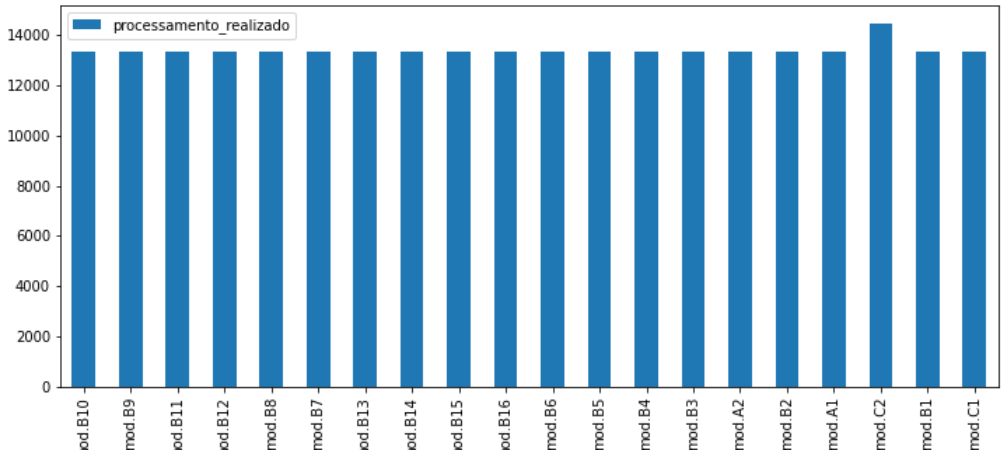


Figura 10 – Processamento realizado por cada uma das 20 máquinas com Dynamic FPLTF

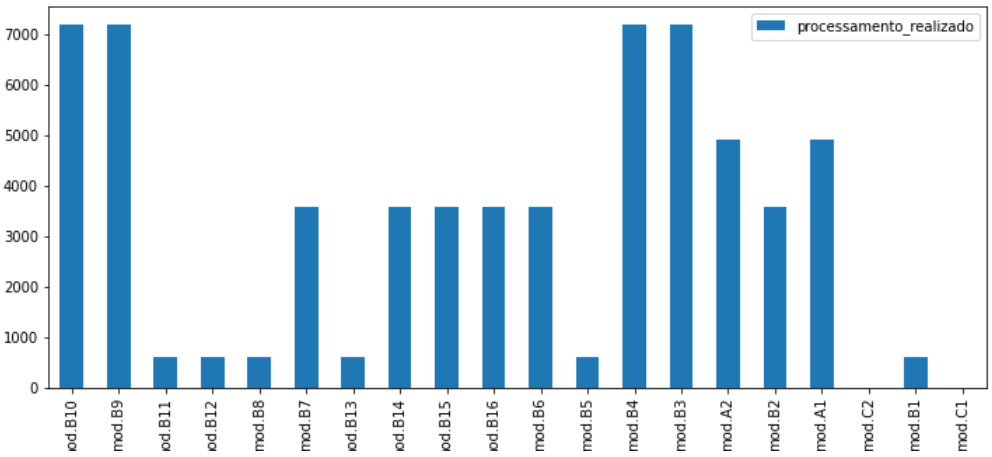


Figura 11 – Processamento realizado por cada uma das 20 máquinas com Sufferage

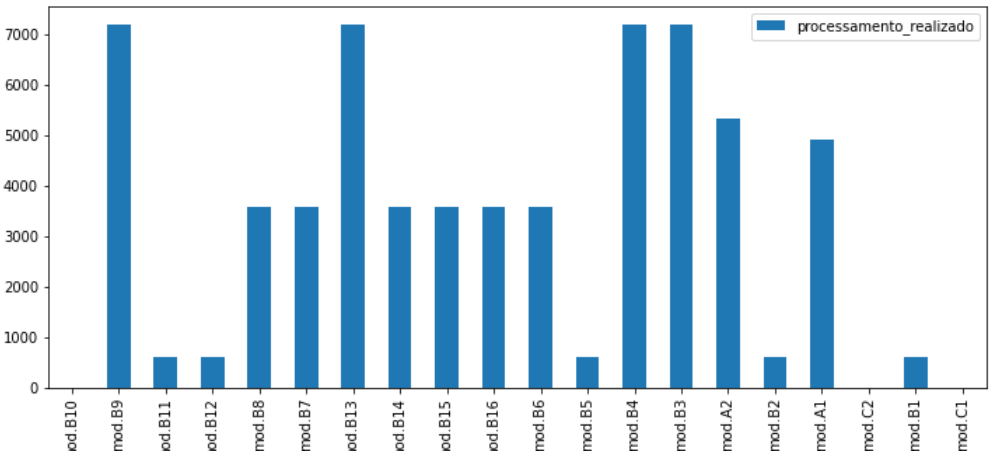


Figura 12 – Processamento realizado por cada uma das 20 máquinas com Min-min

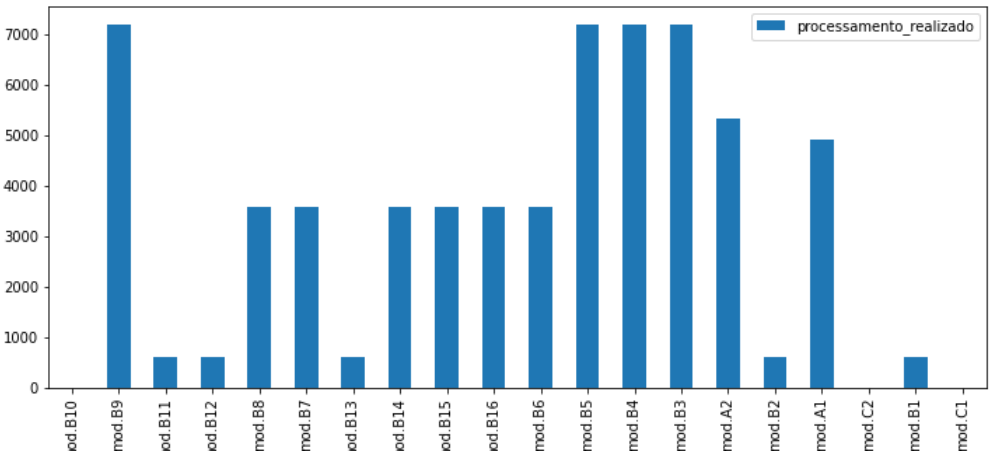


Figura 13 – Processamento realizado por cada uma das 20 máquinas com Max-min

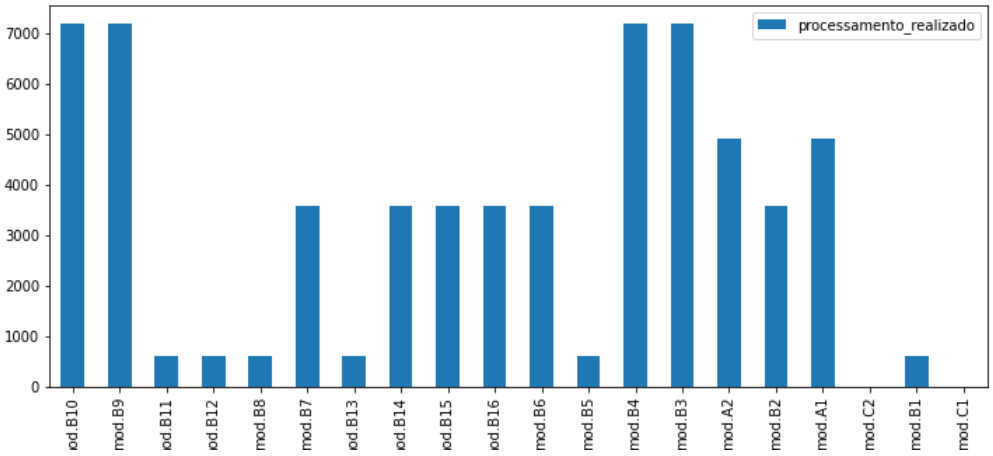
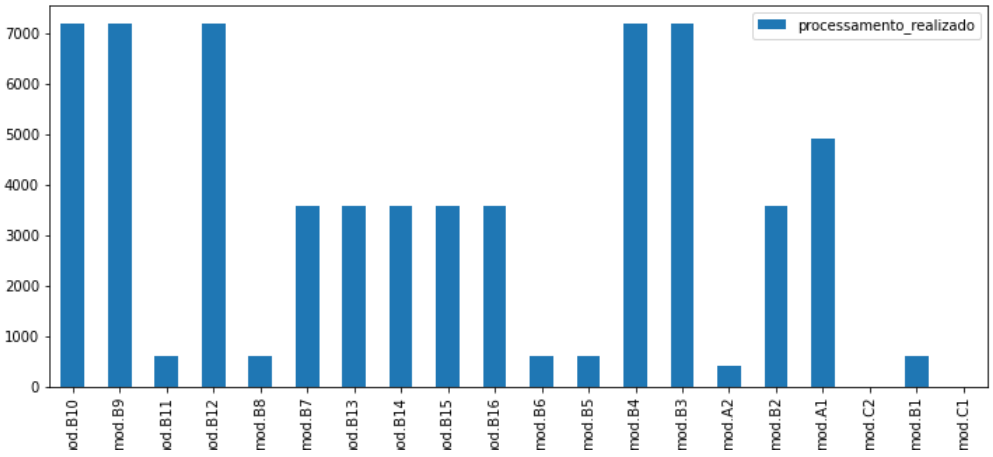


Figura 14 – Processamento realizado por cada uma das 20 máquinas com HOSEP



A.3 40 máquinas

Seguem os gráficos de tempo de ocupação gerados para testes com 40 máquinas.

Figura 15 – Processamento realizado por cada uma das 40 máquinas com Round-Robin

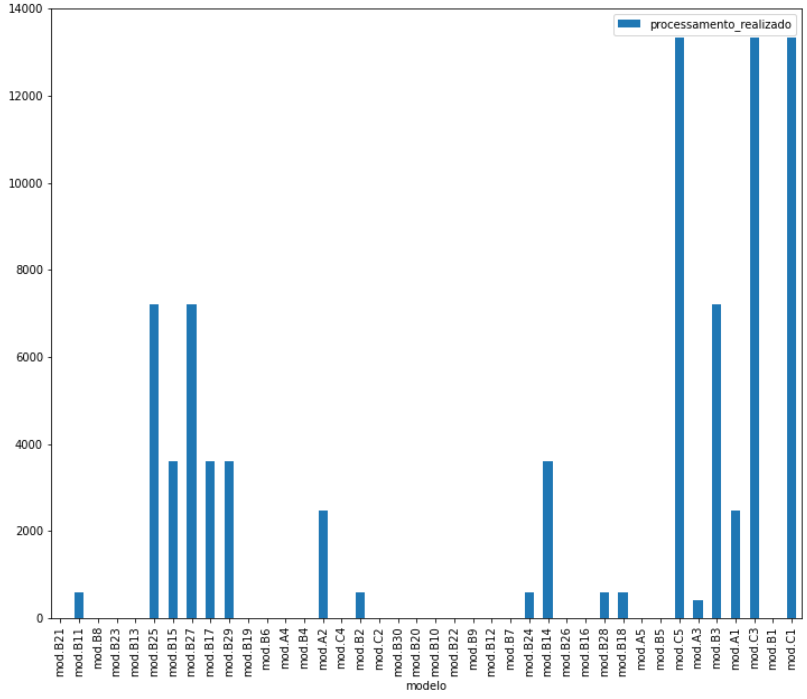


Figura 16 – Processamento realizado por cada uma das 40 máquinas com Workqueue with replication

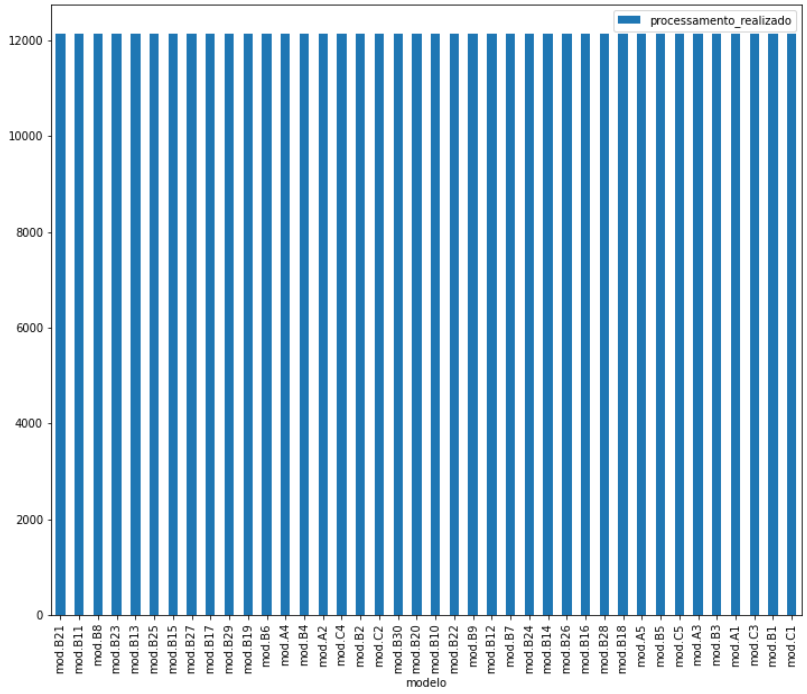


Figura 17 – Processamento realizado por cada uma das 40 máquinas com Dynamic FPLTF

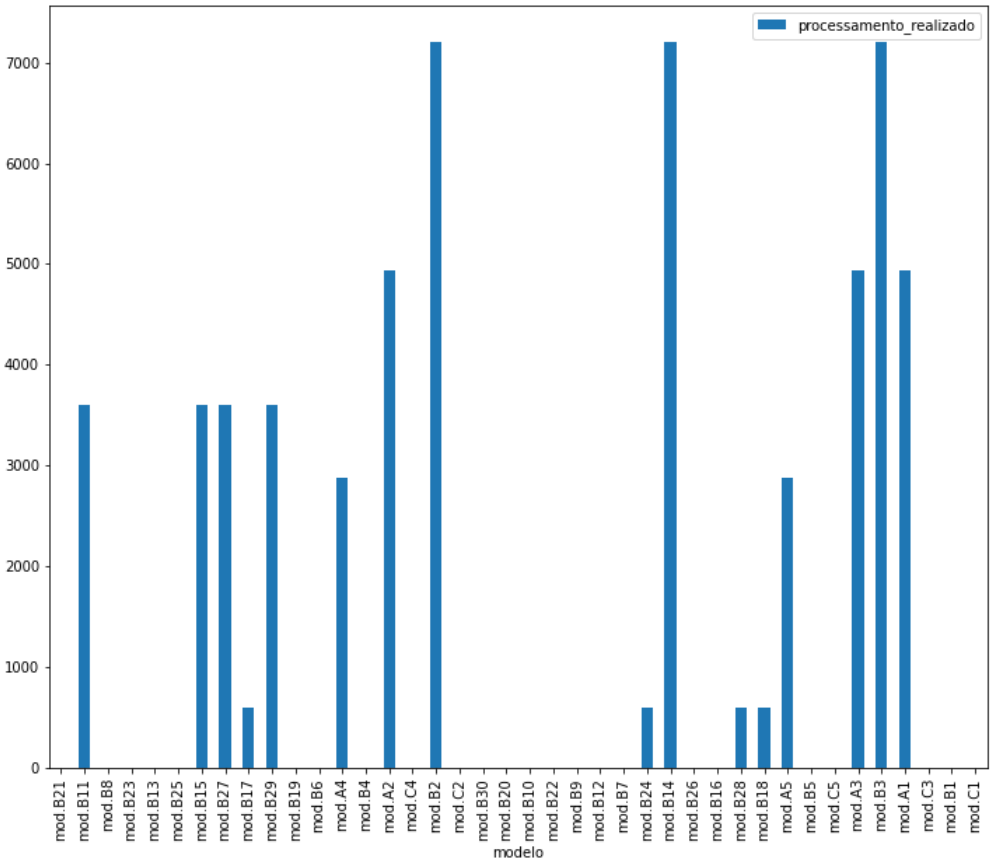


Figura 18 – Processamento realizado por cada uma das 40 máquinas com Sufferage

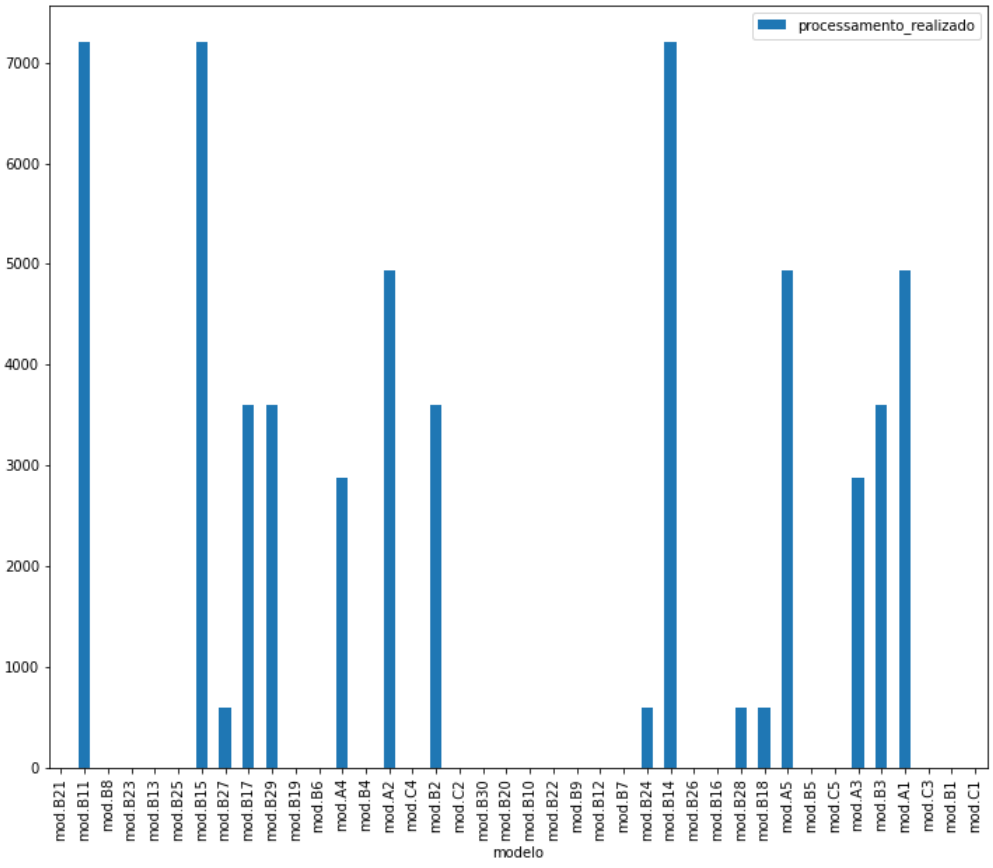


Figura 19 – Processamento realizado por cada uma das 40 máquinas com Min-min

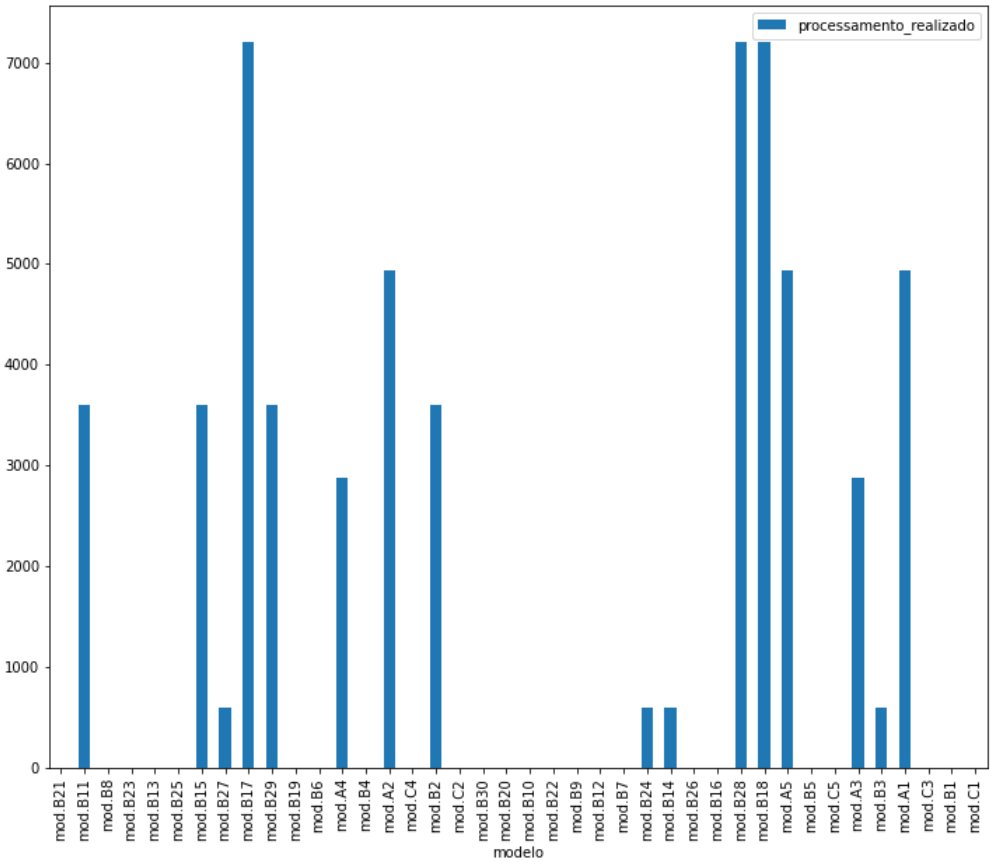


Figura 20 – Processamento realizado por cada uma das 40 máquinas com Max-min

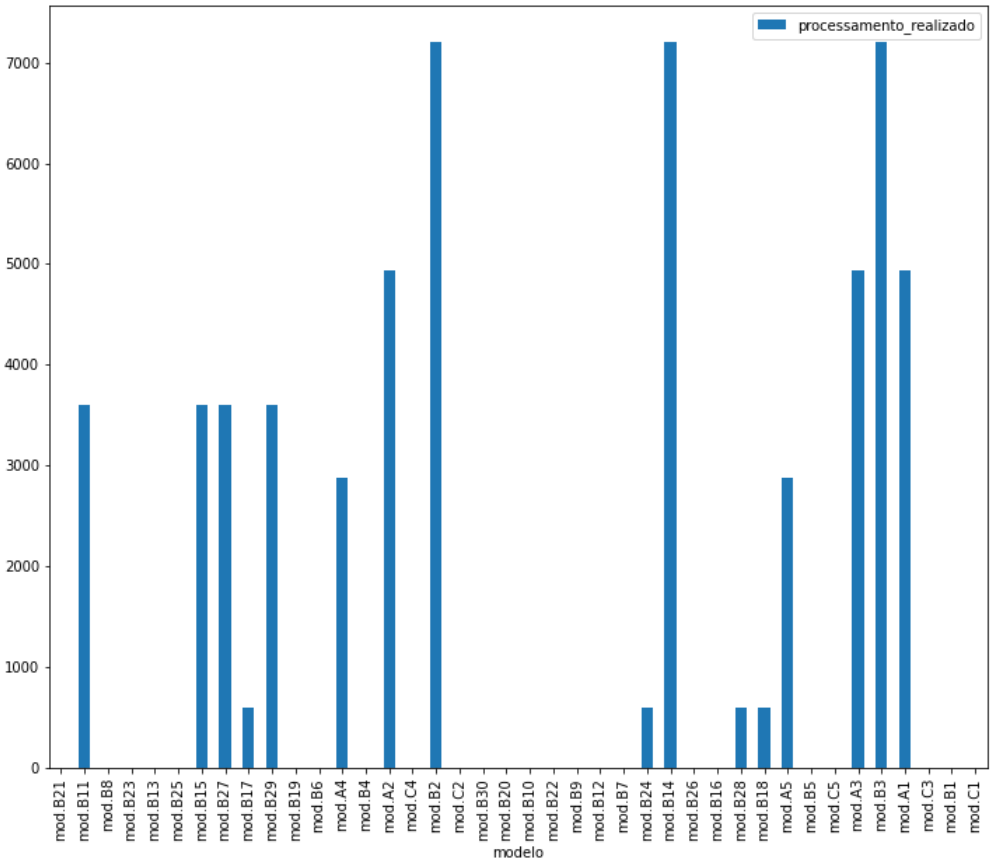
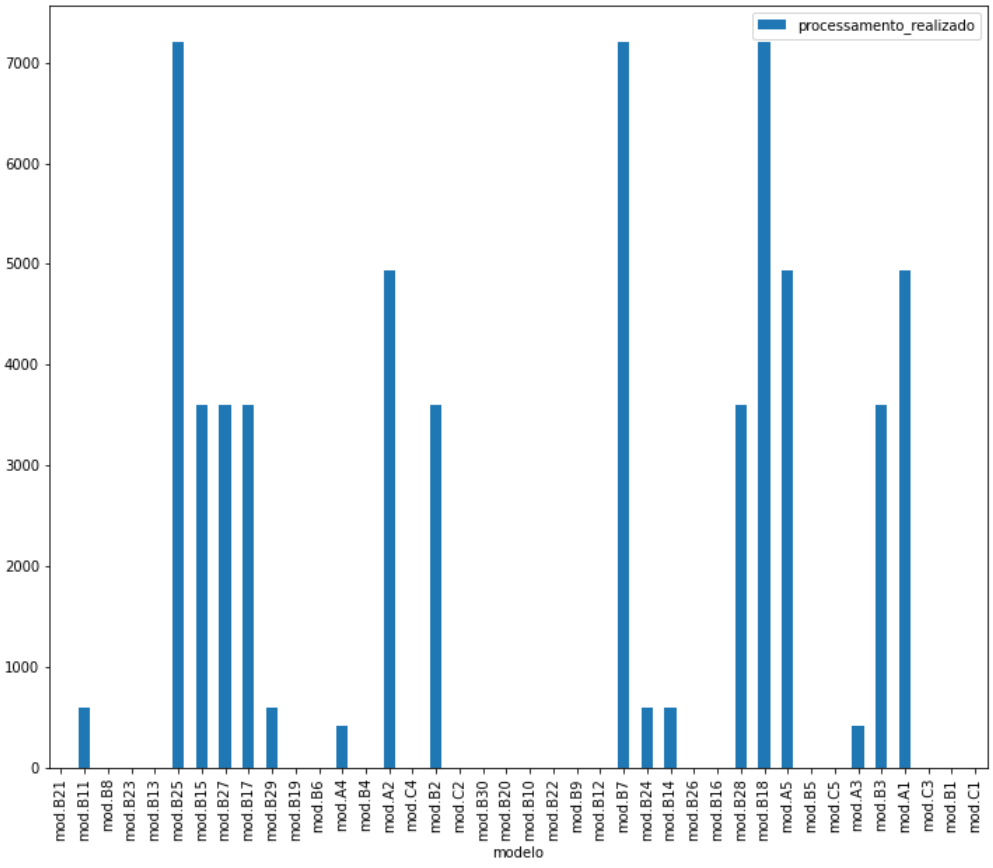


Figura 21 – Processamento realizado por cada uma das 40 máquinas com HOSEP



APÊNDICE B – 180 tarefas uniformes - Processamento em cada nó

Neste apêndice é apresentado, na forma de gráficos, o processamento realizado por cada máquina na simulação de 180 tarefas uniformes.

B.1 10 máquinas

Seguem os gráficos de tempo de ocupação gerados para testes com 10 máquinas.

Figura 22 – Processamento realizado por cada máquina com Round-Robin

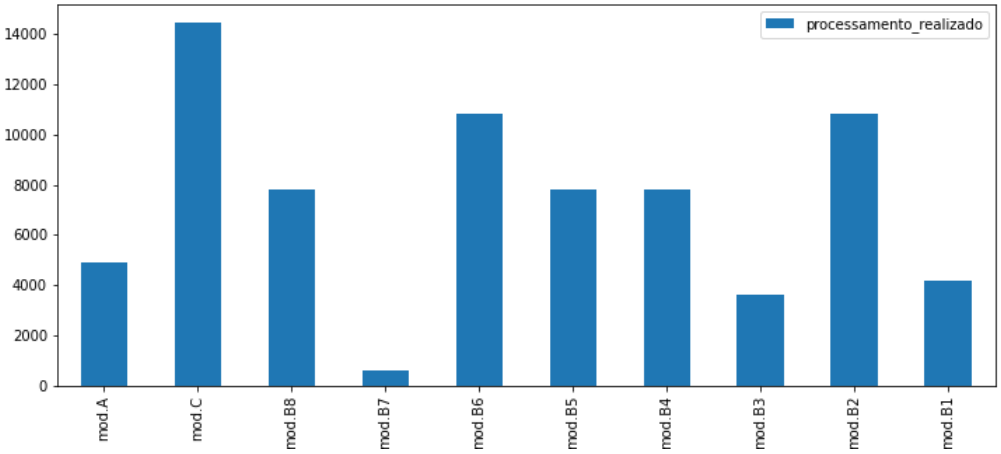


Figura 23 – Processamento realizado por cada uma das 10 máquinas com Workqueue with replication

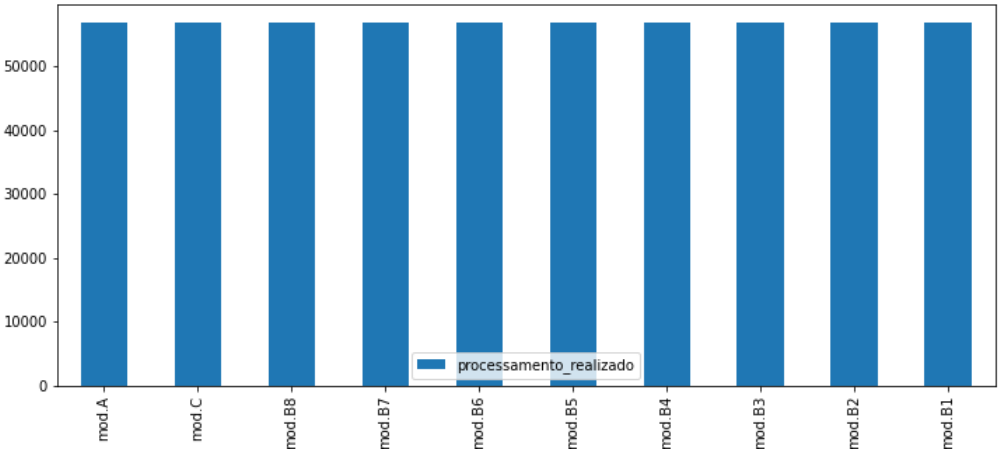


Figura 24 – Processamento realizado por cada uma das 10 máquinas com Dynamic FPLTF

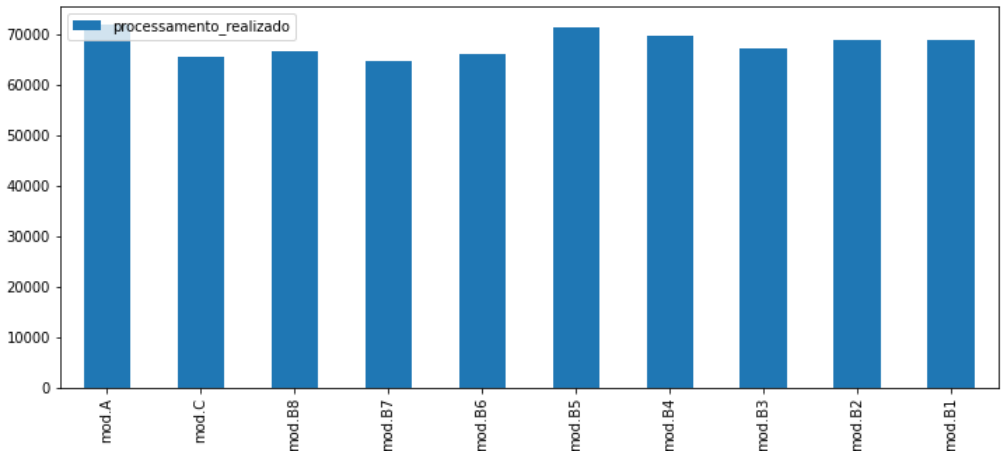


Figura 25 – Processamento realizado por cada uma das 10 máquinas com Sufferage

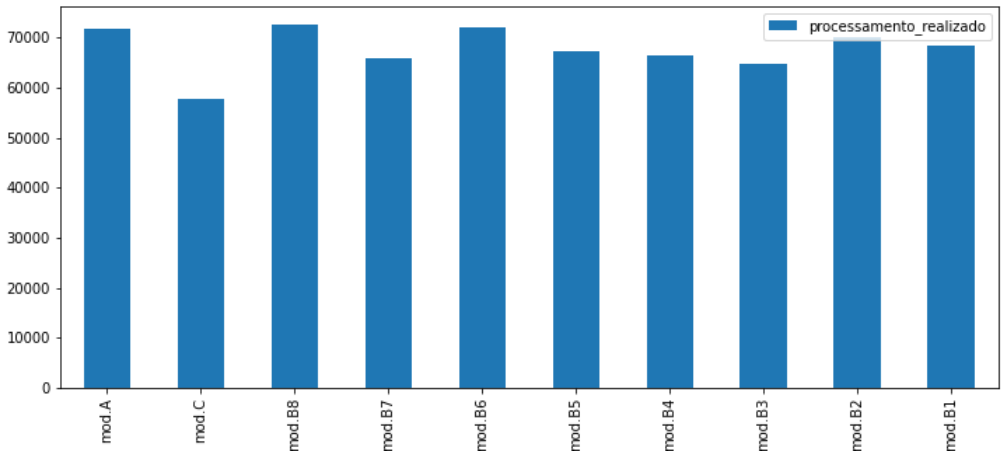


Figura 26 – Processamento realizado por cada uma das 10 máquinas com Min-min

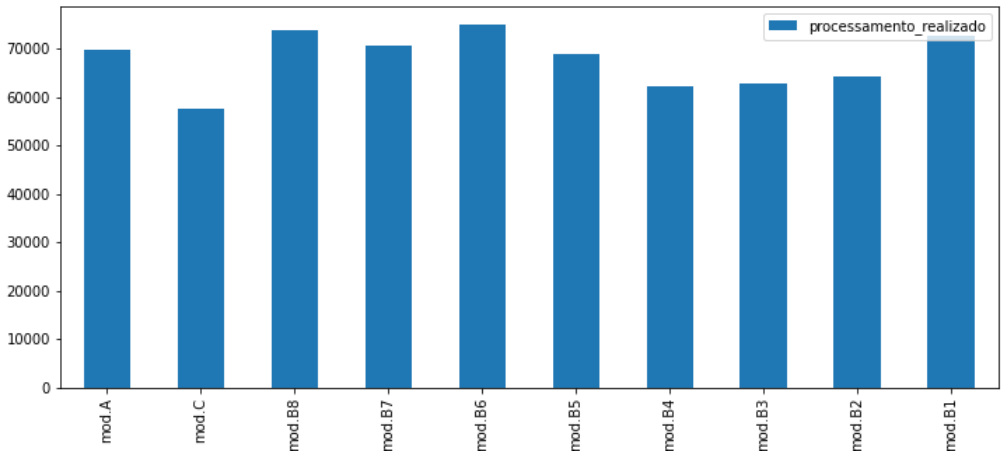


Figura 27 – Processamento realizado por cada uma das 10 máquinas com Max-min

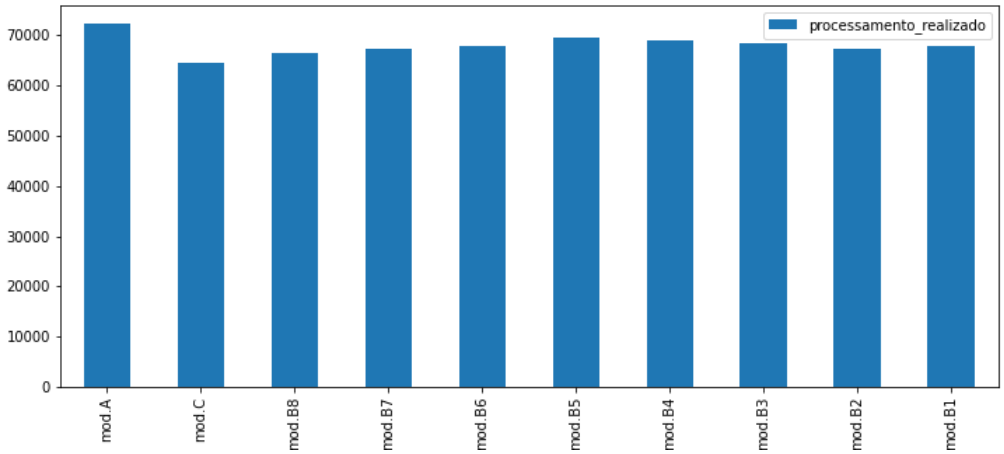
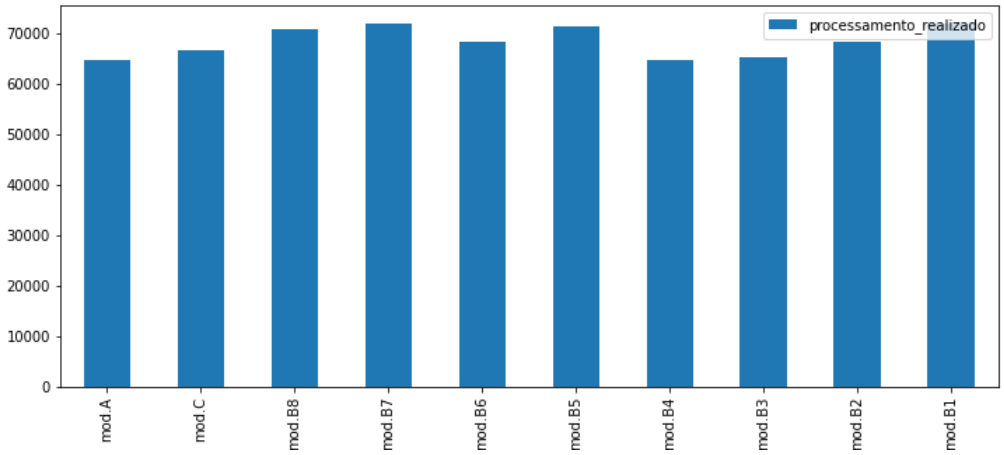


Figura 28 – Processamento realizado por cada uma das 10 máquinas com HOSEP



B.2 20 máquinas

Seguem os gráficos de tempo de ocupação gerados para testes com 20 máquinas.

Figura 29 – Processamento realizado por cada uma das 20 máquinas com Round-Robin

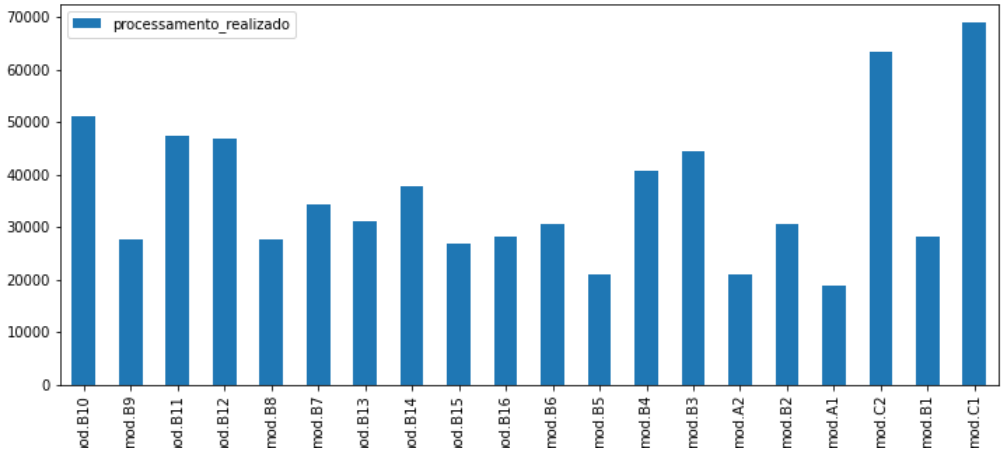


Figura 30 – Processamento realizado por cada uma das 20 máquinas com Workqueue with replication

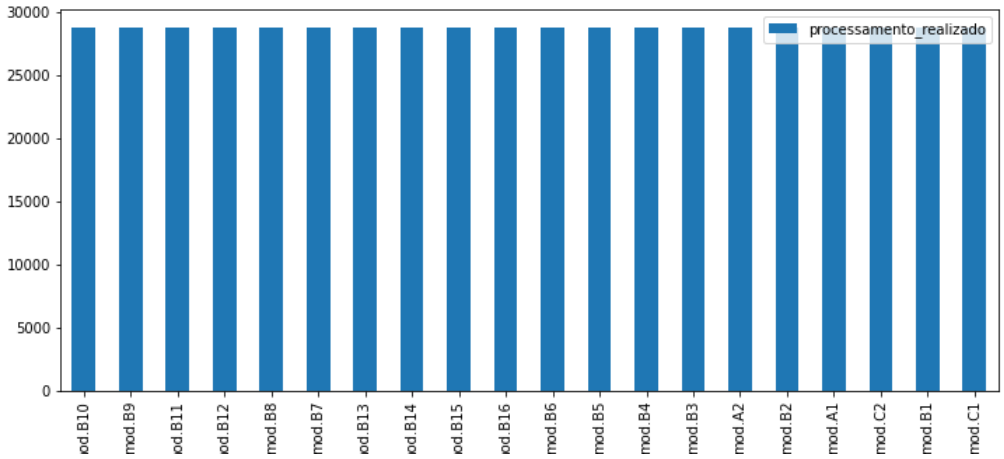


Figura 31 – Processamento realizado por cada uma das 20 máquinas com Dynamic FPLTF

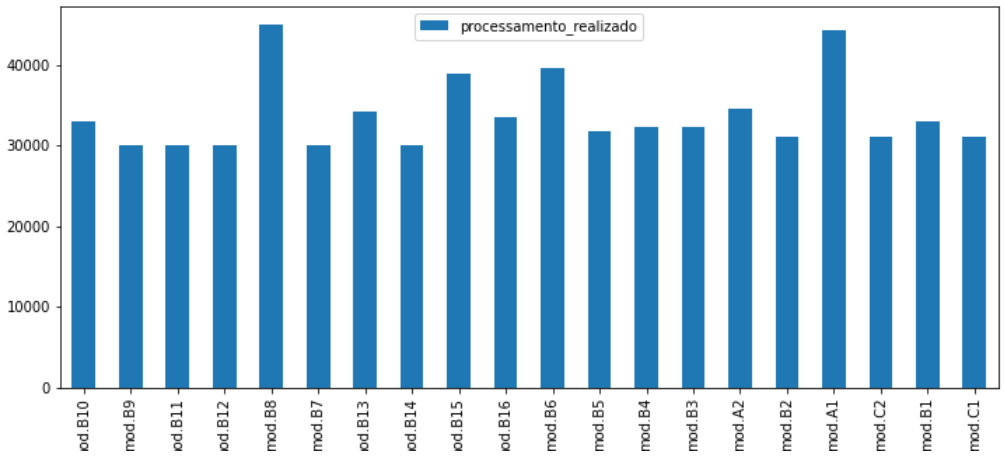


Figura 32 – Processamento realizado por cada uma das 20 máquinas com Sufferage

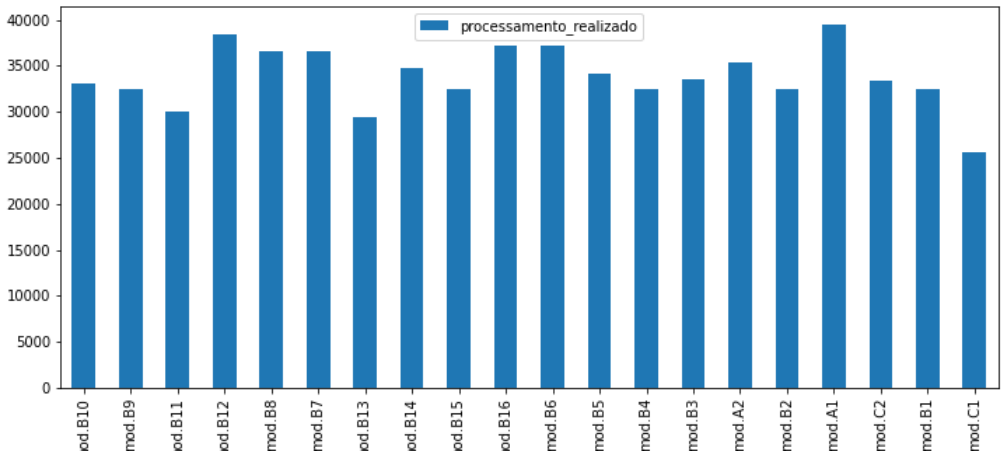


Figura 33 – Processamento realizado por cada uma das 20 máquinas com Min-min

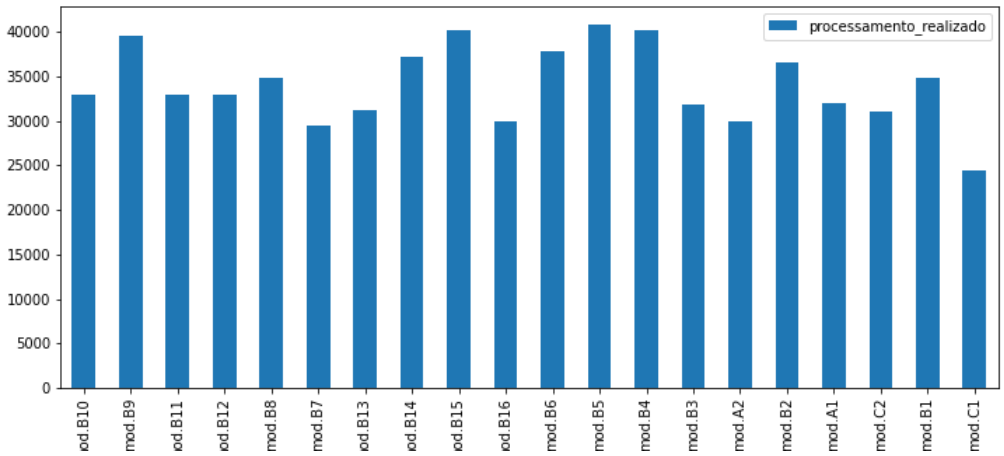


Figura 34 – Processamento realizado por cada uma das 20 máquinas com Max-min

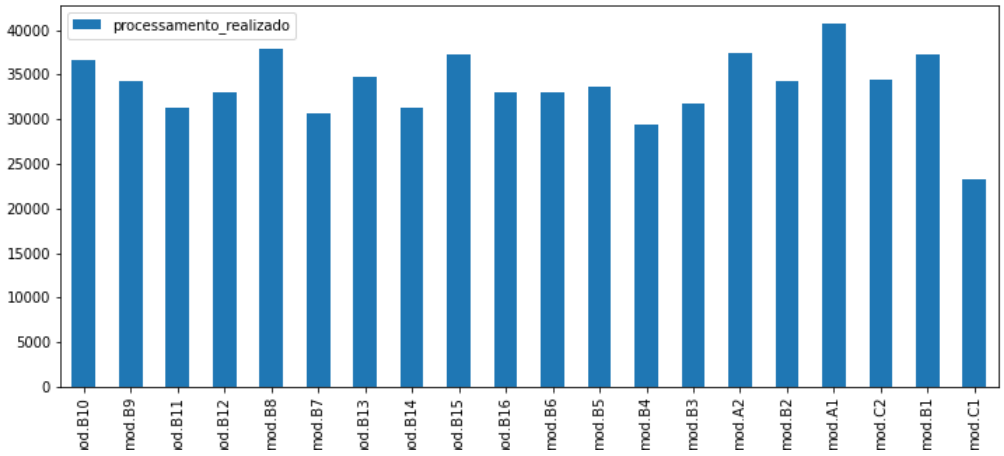
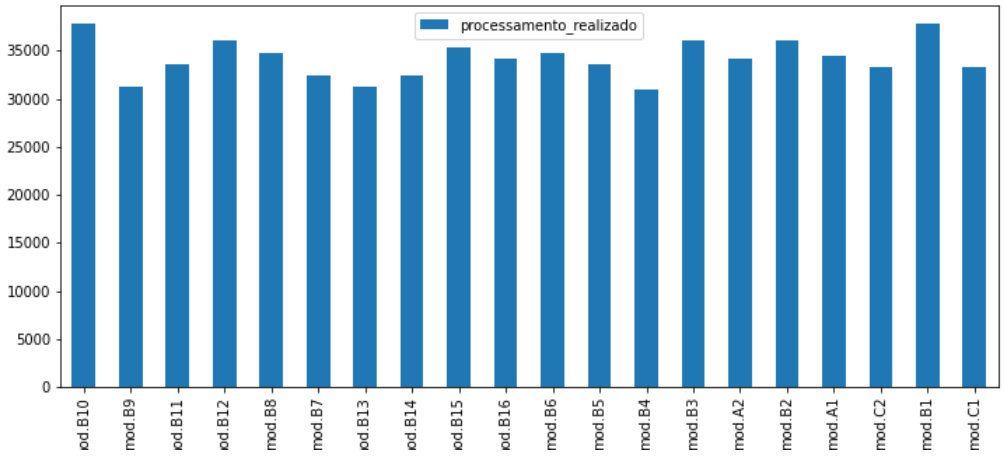


Figura 35 – Processamento realizado por cada uma das 20 máquinas com HOSEP



B.3 40 máquinas

Seguem os gráficos de tempo de ocupação gerados para testes com 40 máquinas.

Figura 36 – Processamento realizado por cada uma das 40 máquinas com Round-Robin

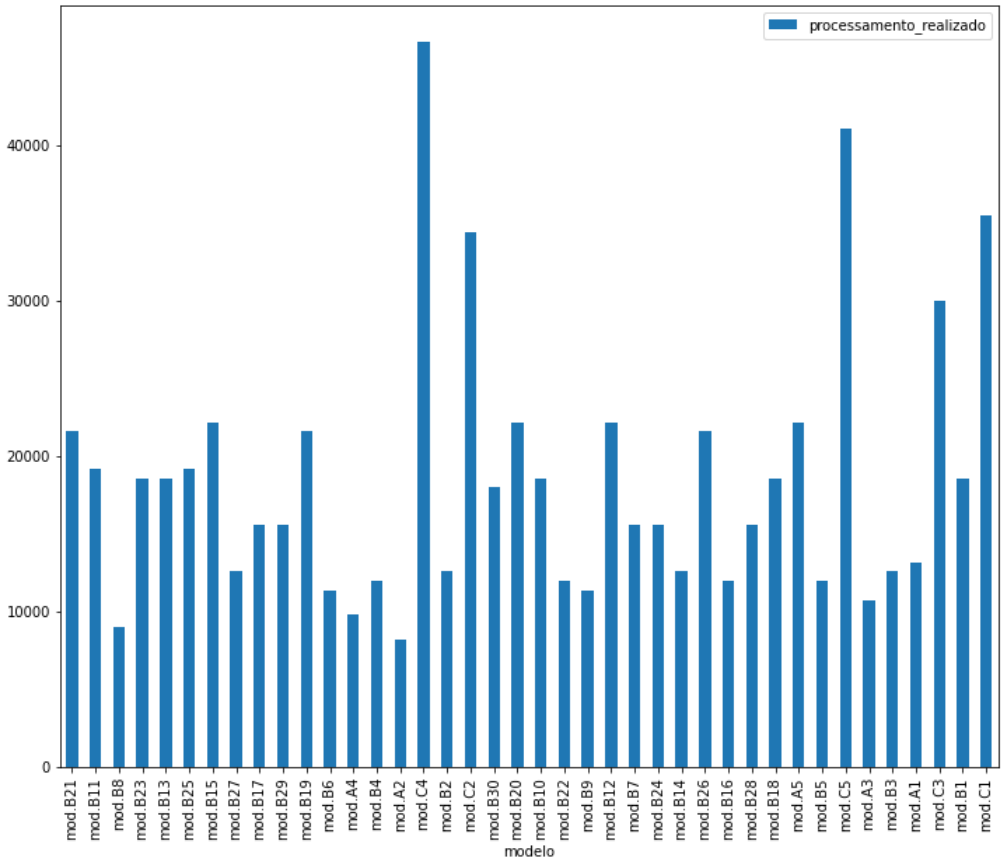


Figura 37 – Processamento realizado por cada uma das 40 máquinas com Workqueue with replication

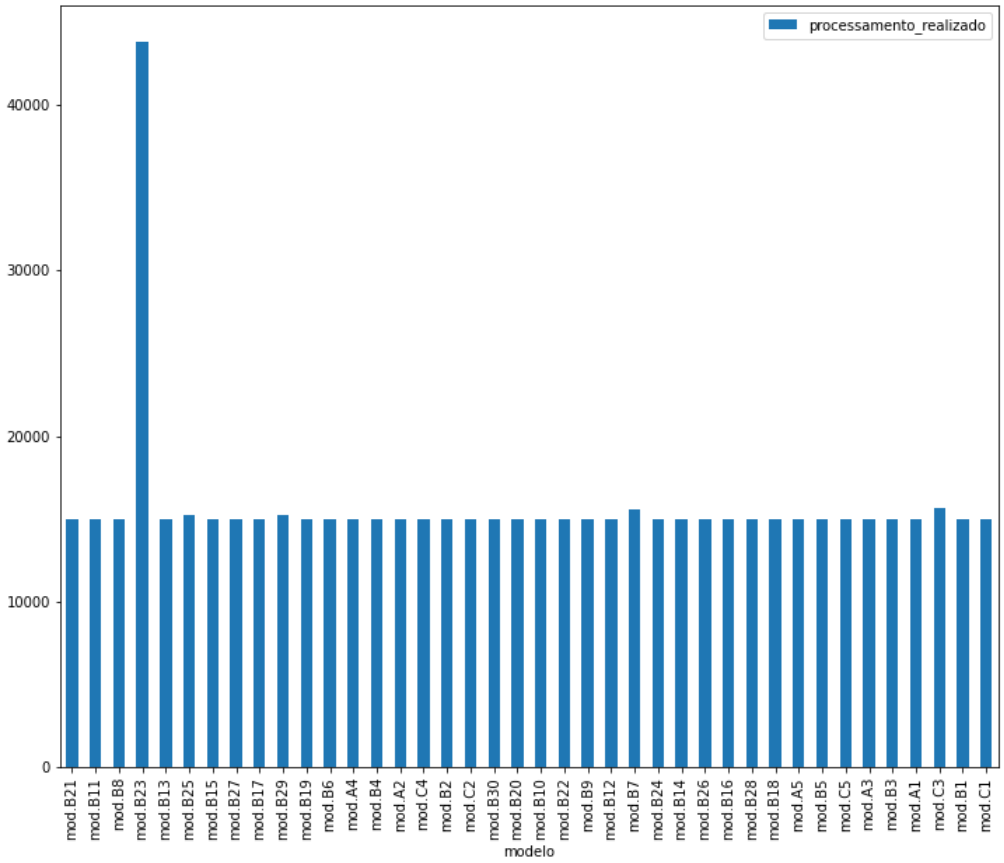


Figura 38 – Processamento realizado por cada uma das 40 máquinas com Dynamic FPLTF

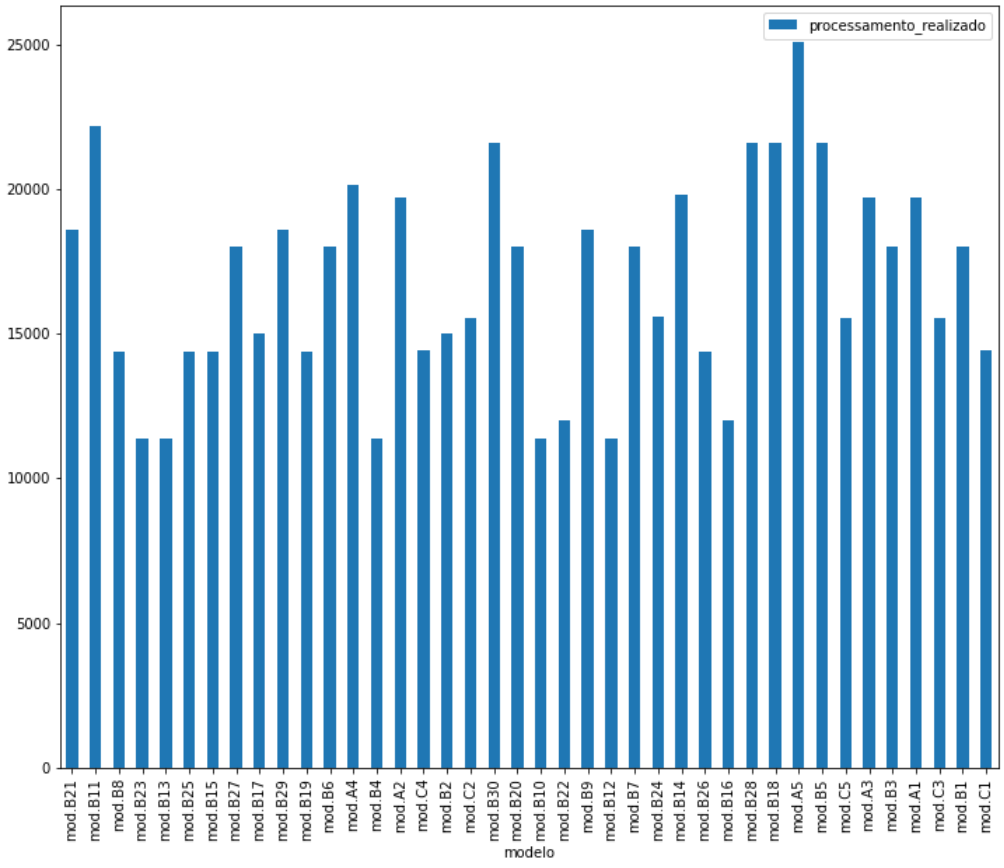


Figura 39 – Processamento realizado por cada uma das 40 máquinas com Sufferage

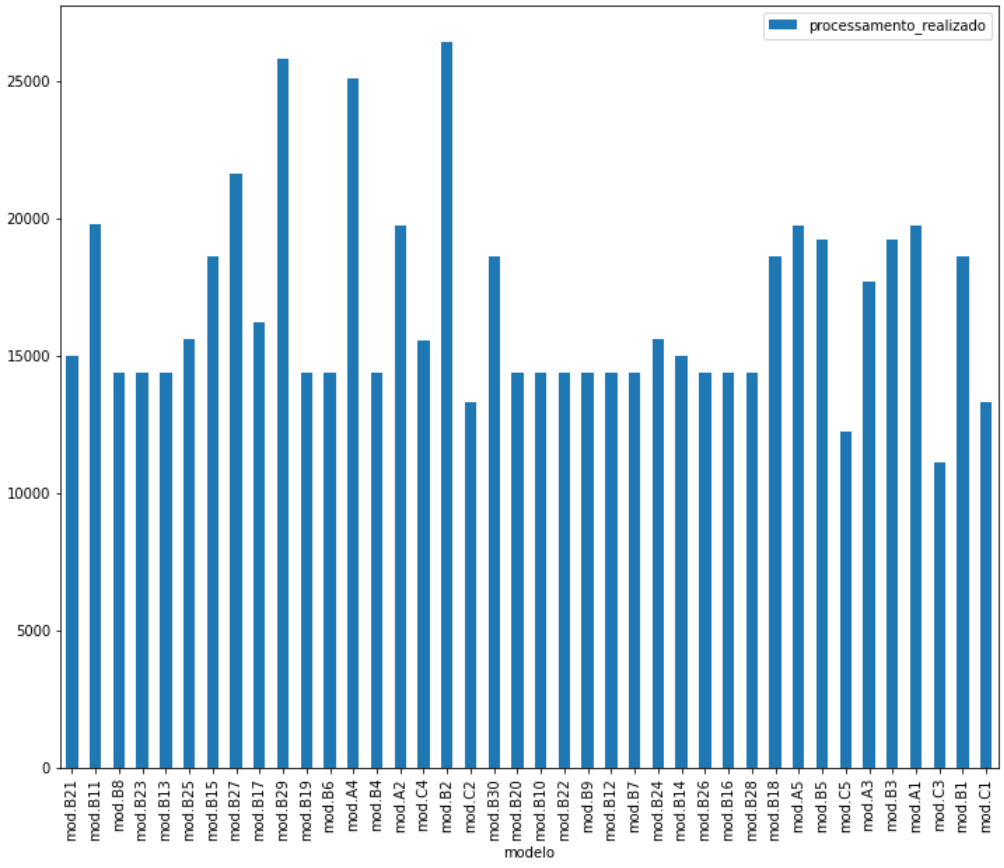


Figura 40 – Processamento realizado por cada uma das 40 máquinas com Min-min

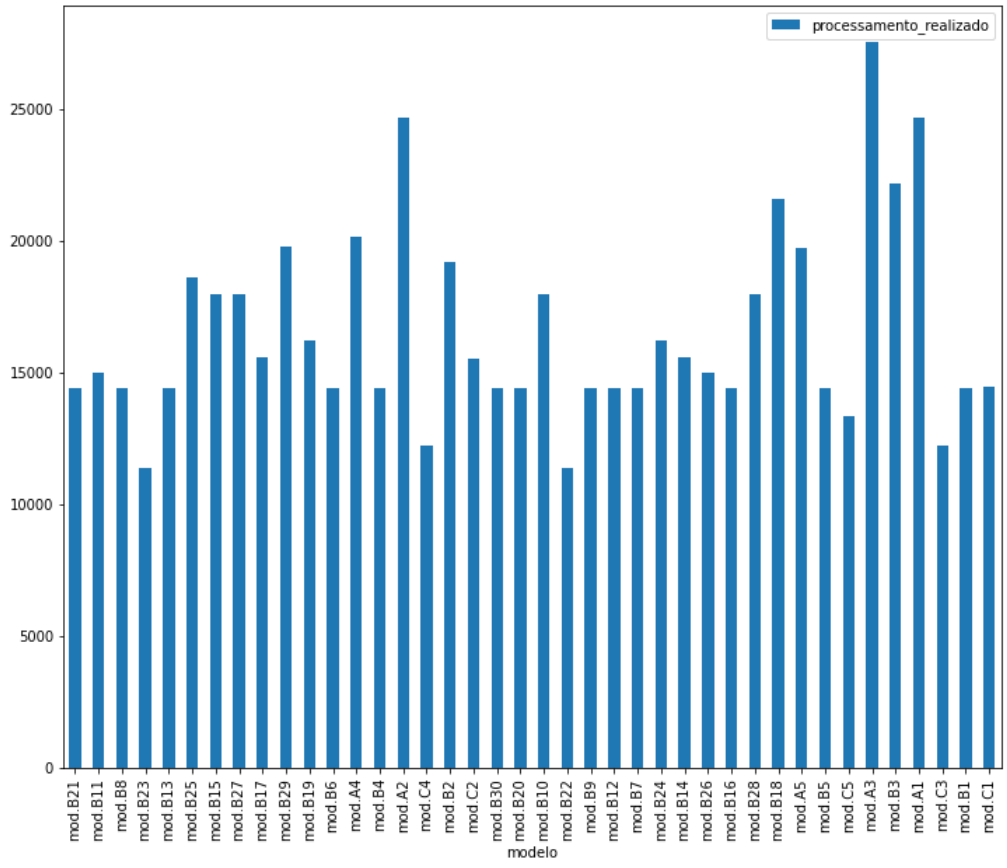


Figura 41 – Processamento realizado por cada uma das 40 máquinas com Max-min

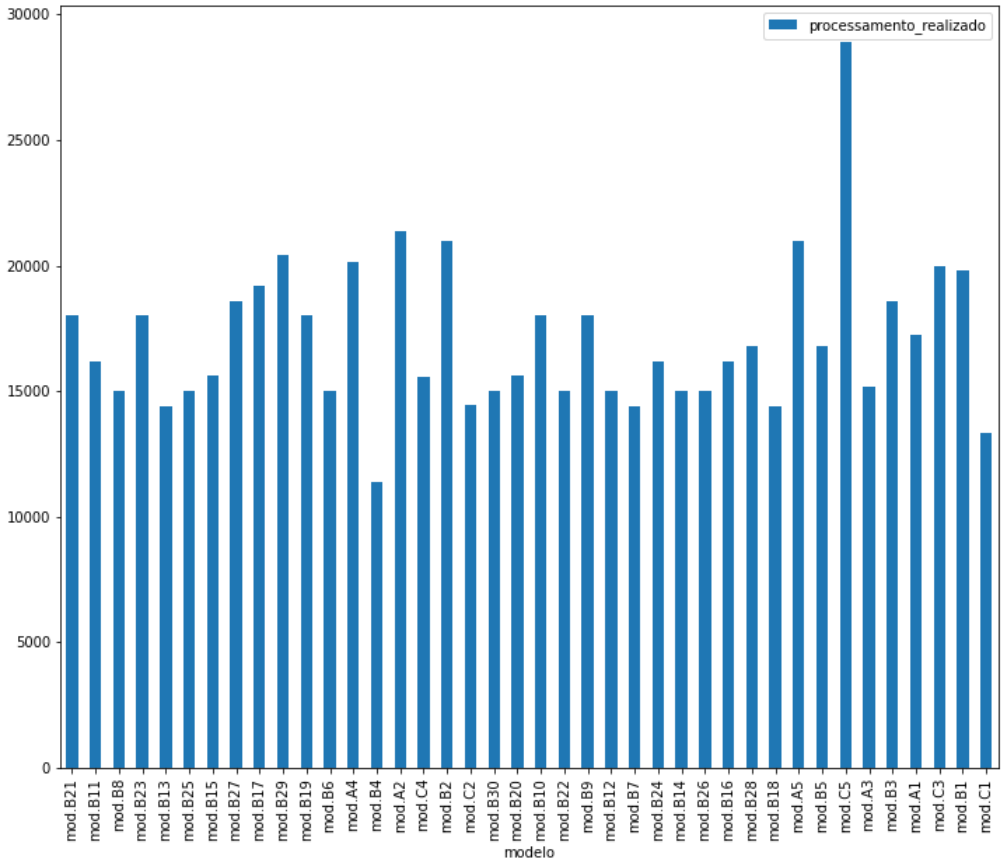
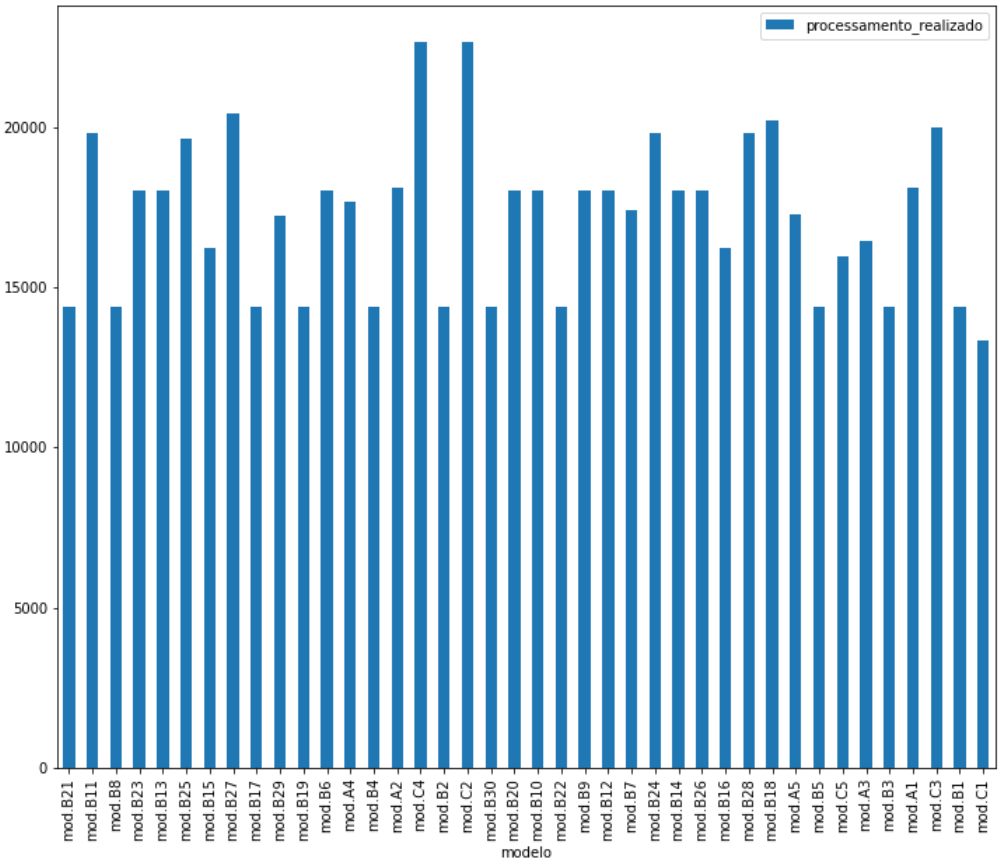


Figura 42 – Processamento realizado por cada uma das 40 máquinas com HOSEP



APÊNDICE C – Poucas tarefas grandes e muitas tarefas pequenas - Processamento em cada nó

Neste apêndice é apresentado, na forma de gráficos, o processamento realizado por cada máquina na simulação de poucas tarefas grandes e muitas tarefas pequenas.

Esta simulação foi executada apenas com a grade de 10 máquinas.

Figura 43 – Processamento realizado por cada uma das 10 máquinas com Round-Robin

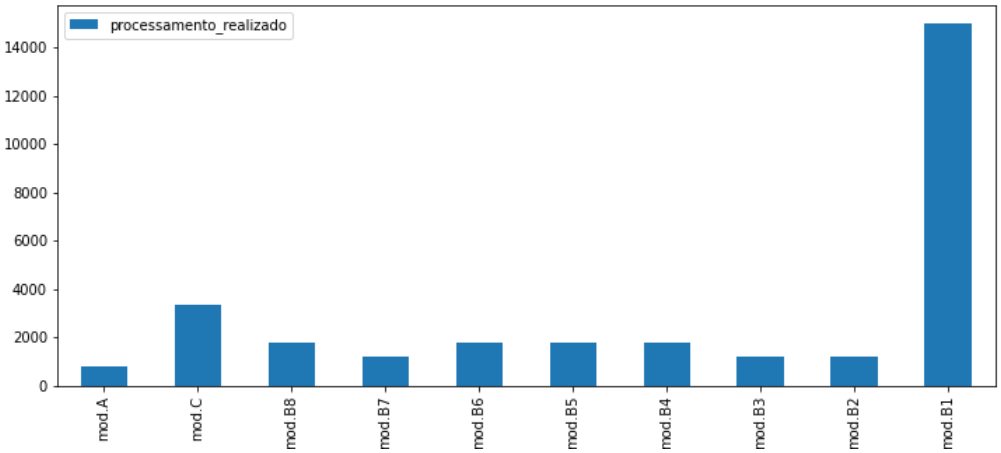


Figura 44 – Processamento realizado por cada uma das 10 máquinas com Workqueue with replication

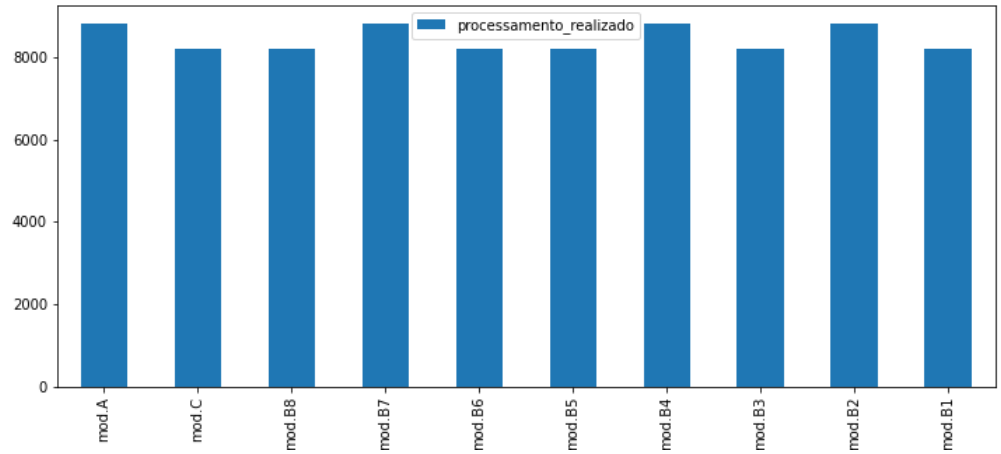


Figura 45 – Processamento realizado por cada uma das 10 máquinas com Dynamic FPLTF

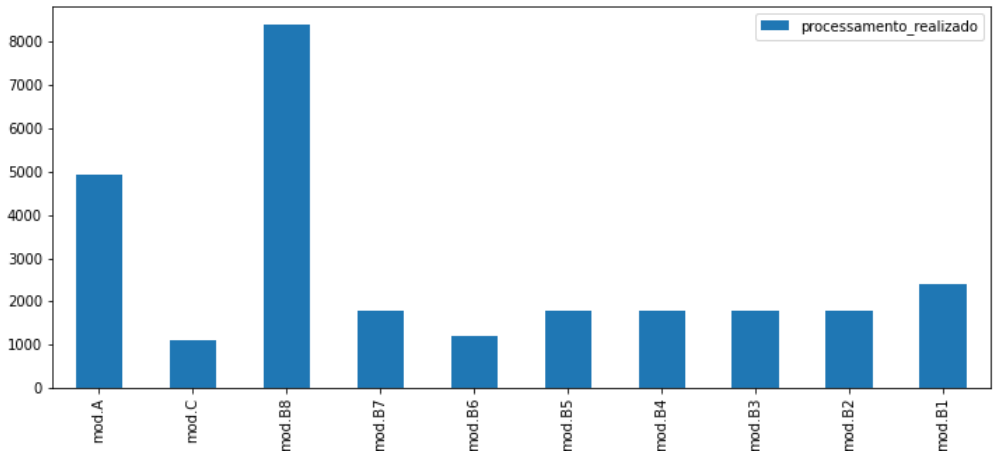


Figura 46 – Processamento realizado por cada uma das 10 máquinas com Sufferage

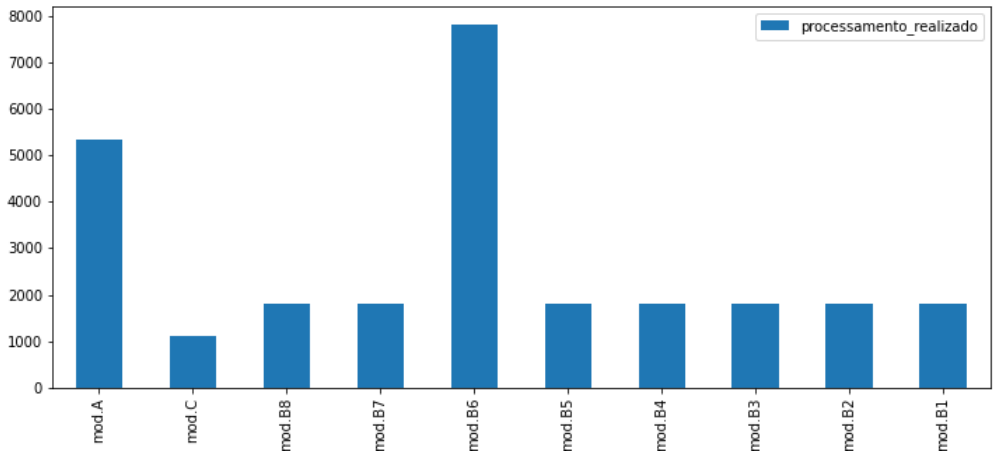


Figura 47 – Processamento realizado por cada uma das 10 máquinas com Min-min

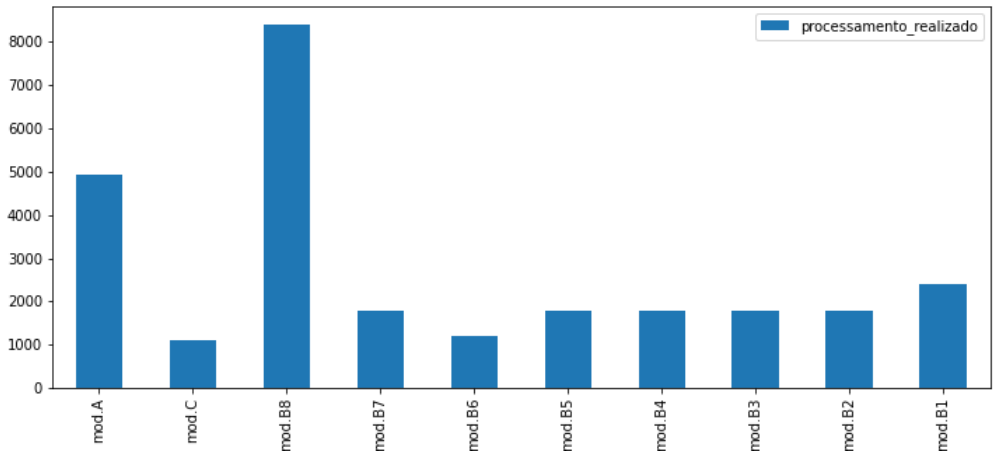


Figura 48 – Processamento realizado por cada uma das 10 máquinas com Max-min

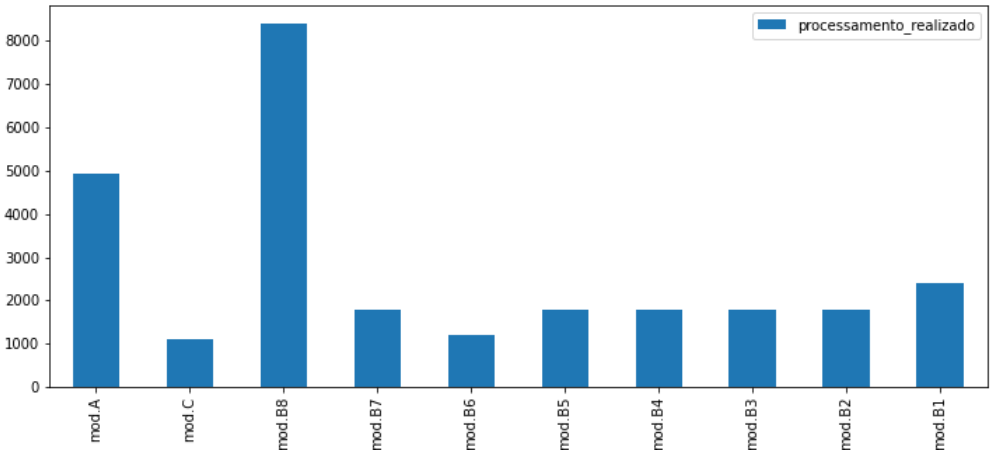
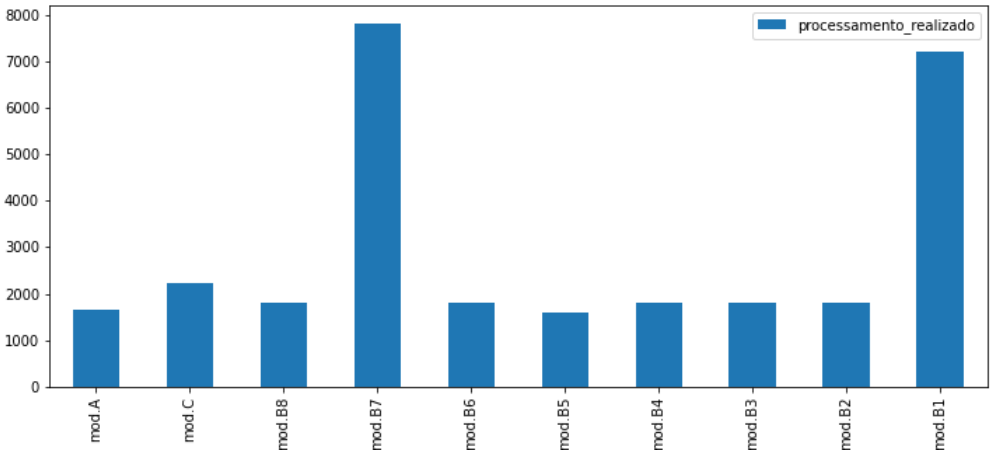


Figura 49 – Processamento realizado por cada uma das 10 máquinas com HOSEP



APÊNDICE D – 30 tarefas grandes - Processamento em cada nó

Neste apêndice é apresentado, na forma de gráficos, o processamento realizado por cada máquina na simulação de 30 tarefas grandes.

Esta simulação foi executada apenas com a grade de 10 máquinas.

Figura 50 – Processamento realizado por cada uma das 10 máquinas com Round-Robin

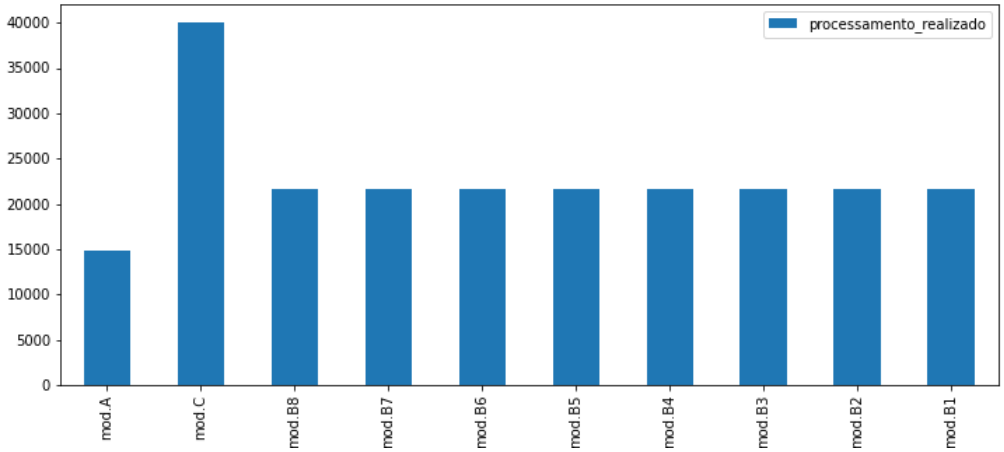


Figura 51 – Processamento realizado por cada uma das 10 máquinas com Workqueue with replication

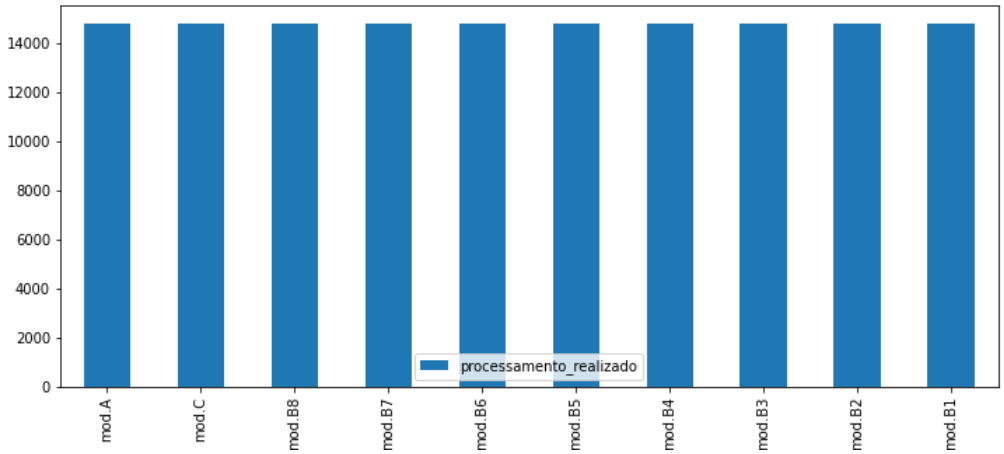


Figura 52 – Processamento realizado por cada uma das 10 máquinas com Dynamic FPLTF

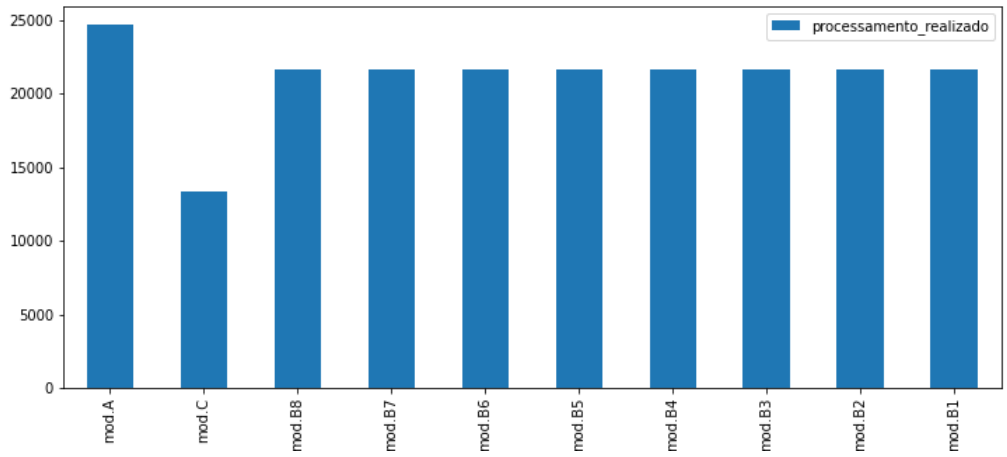


Figura 53 – Processamento realizado por cada uma das 10 máquinas com Sufferage

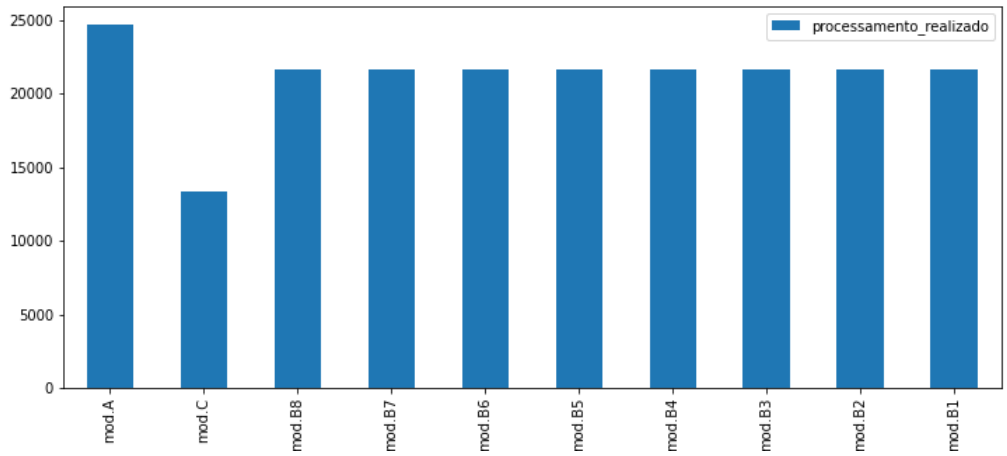


Figura 54 – Processamento realizado por cada uma das 10 máquinas com Min-min

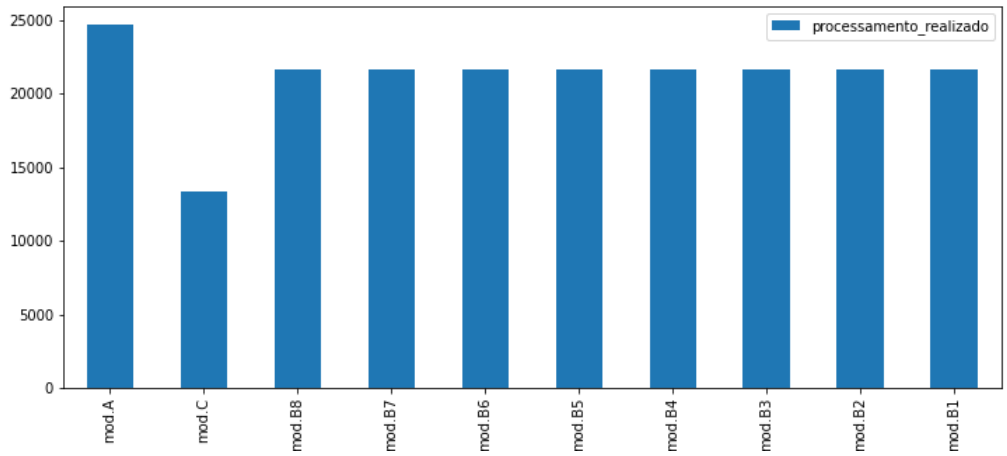


Figura 55 – Processamento realizado por cada uma das 10 máquinas com Max-min

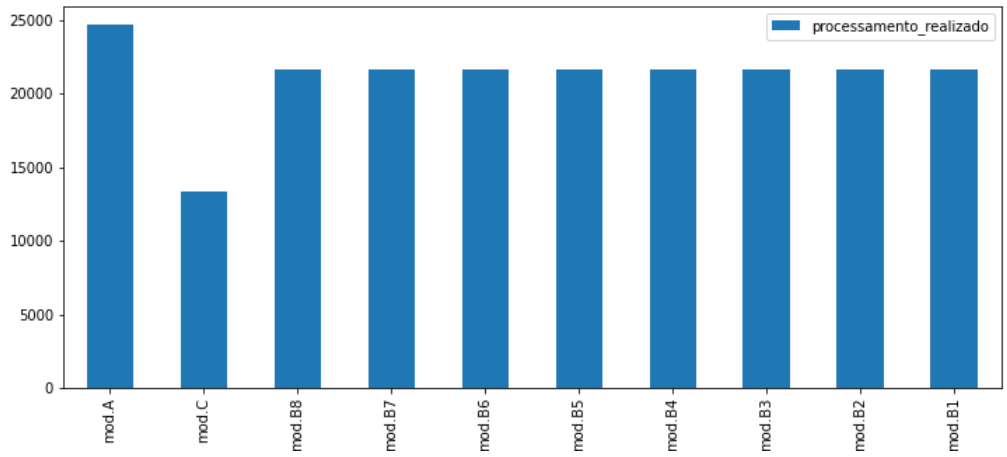


Figura 56 – Processamento realizado por cada uma das 10 máquinas com HOSEP

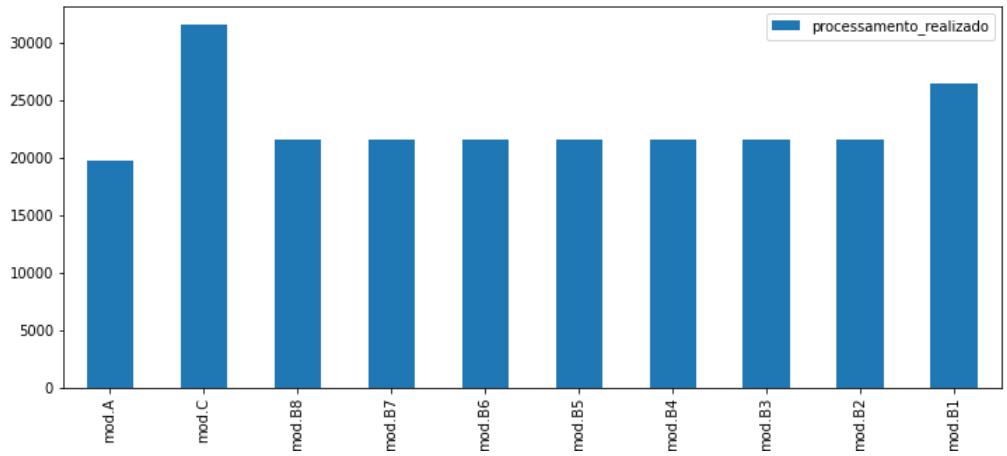
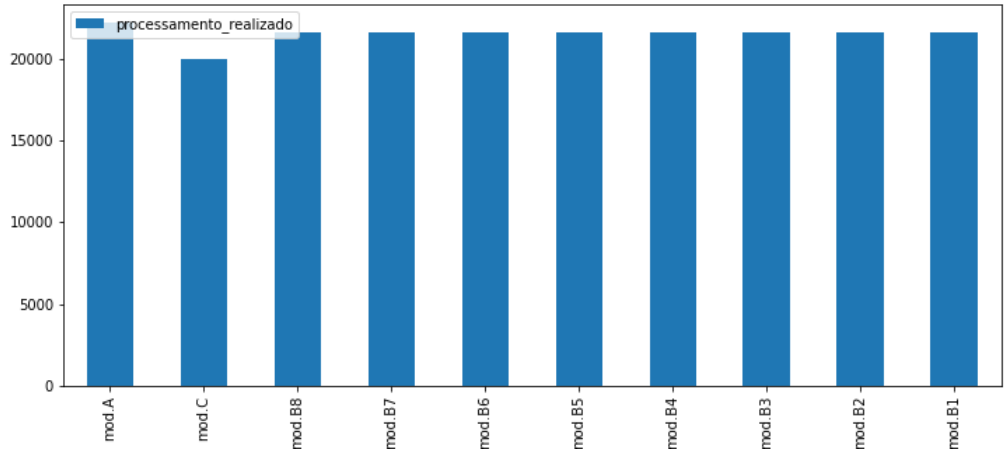


Figura 57 – Processamento realizado por cada máquina com Dynamic FPLTF



APÊNDICE E – 360 tarefas curtas - Processamento em cada nó

Neste apêndice é apresentado, na forma de gráficos, o processamento realizado por cada máquina na simulação de 360 tarefas curtas.

Esta simulação foi executada apenas com a grade de 10 máquinas.

Figura 58 – Processamento realizado por cada uma das 10 máquinas com Round-Robin

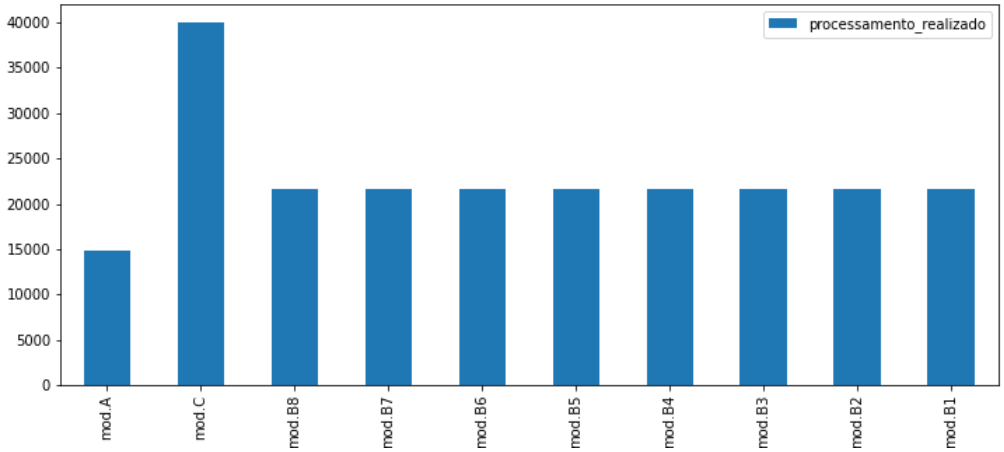


Figura 59 – Processamento realizado por cada uma das 10 máquinas com Workqueue with replication

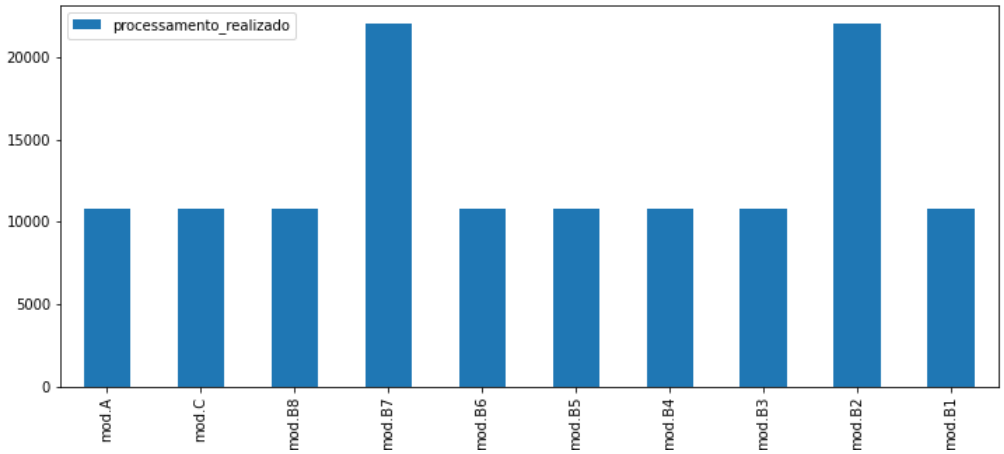


Figura 60 – Processamento realizado por cada uma das 10 máquinas com Sufferage

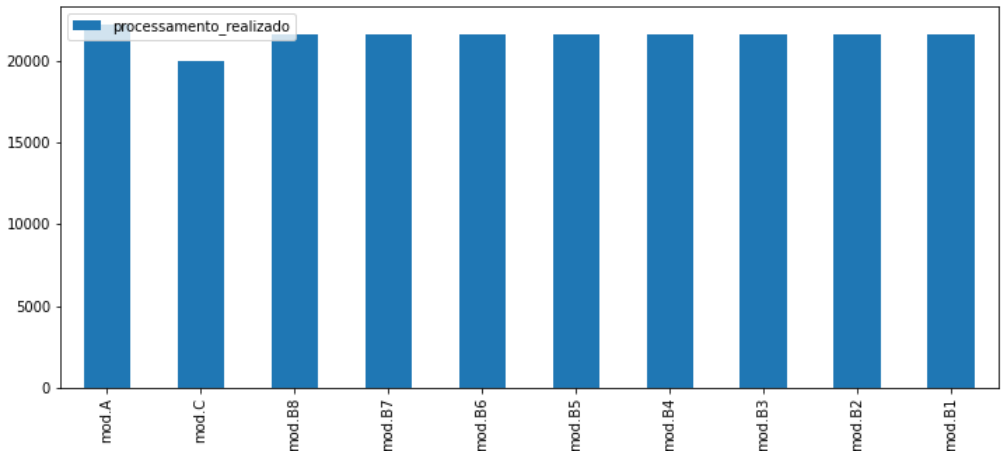


Figura 61 – Processamento realizado por cada uma das 10 máquinas com Min-min

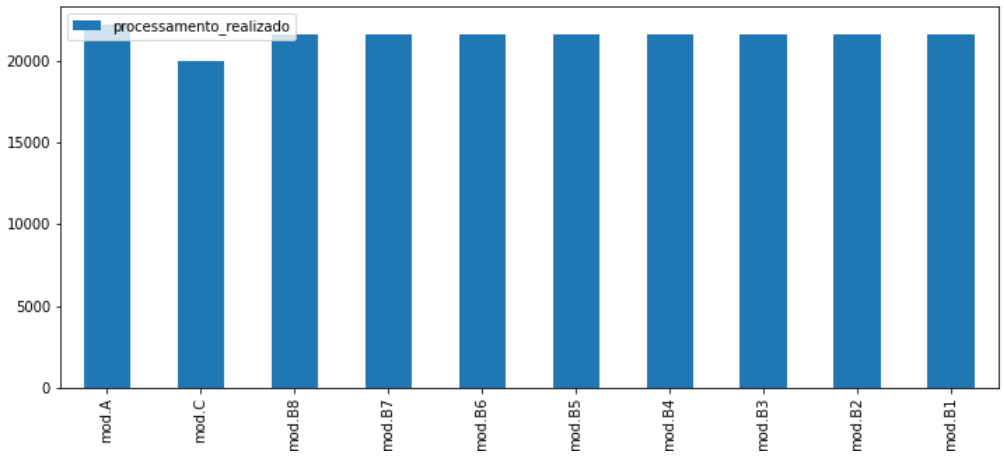


Figura 62 – Processamento realizado por cada uma das 10 máquinas com Max-min

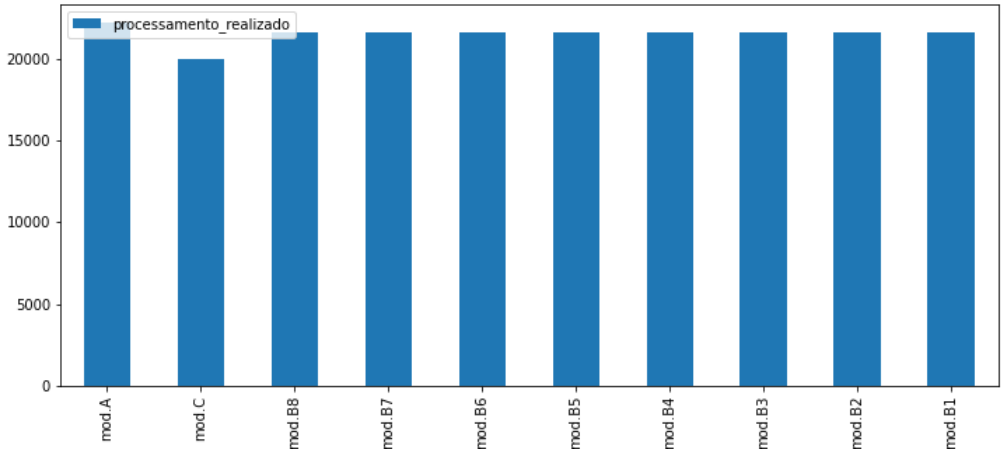


Figura 63 – Processamento realizado por cada uma das 10 máquinas com HOSEP