DOI: 10.18605/2175-7275/cereus.v10n2p26-36

<<Recebido: 05/06/2018. Aceito: 03/07/2018 >>

ARTIGO ORIGINAL

REVISTA CEREUS ISSN: 2175-7275

# Gerenciador de Processos Distribuídos Pré-Configurado

## *Pre-Configured Distributed Process Manager*

Renê Dettenborn[1], George Lauro Ribeiro de Brito[2], Gentil Veloso Barbosa[3]

**RESUMO**

Este trabalho é direcionado às necessidades emergentes de simulação de sistemas que necessitam diminuir o tempo de processamento com baixo custo de implantação. Problemas relacionados com estudos científicos nas áreas de engenharia, física e biologia e outras que envolvam problemas de simulação e que geram elevado tempo de processamento para alcançar os resultados esperados. Em diversos casos, por mais que se otimize o algoritmo ainda não é possível obter o desempenho desejado. Para resolver este problema, desenvolveu-se um gerenciador de processos distribuídos pré-configurado que engloba o uso da ferramenta Blender e que permite o processamento de grandes massas de dados utilizando qualquer rede de computadores para reduzir o tempo de processamento.

**Palavras-chave**: Simulação de Sistemas, Processos Distribuídos, Pré-configurado, Grml, Blender.

**ABSTRACT**

This work addresses the emerging simulation needs of systems that need to reduce processing time with low deployment costs. Problems related to scientific studies in the areas of engineering, physics and biology and others that involve simulation problems and which generate a high processing time to achieve the expected results. In many cases, even though the algorithm is optimized, it is still not possible to achieve the desired performance. To solve this problem, a preconfigured distributed process manager has been developed that encompasses the use of the Blender tool and allows the processing of large masses of data using any computer network to reduce processing time.

**Keywords**: Systems Simulation, Distributed Processes, Preconfigured, Grml, Blender.

[1] Especialista em Desenvolvimento de Sistemas de Alta Complexidade - Universidade Federal do Tocantins.

E-mail: renedet@gmail.com

[2] Professor Doutor Associado do Curso de Ciência da Computação e do Mestrado em Modelagem Computacional de Sistemas da Universidade Federal do Tocantins.

E-mail: gbrito@uft.edu.br

[3] Professor Doutor Associado do Curso de Ciência da Computação e do Mestrado em Modelagem Computacional de Sistemas da Universidade Federal do Tocantins.

E-mail: gentil@uft.edu.br

## 1. INTRODUÇÃO

Com o passar dos anos a indústria aumentou a capacidade de processamento dos computadores (INTEL, 2018). Mas a necessidade por aumento de processamento computacional continua sendo requisitada em diversas áreas como, por exemplo, em renderização de imagens (LING; GONG, 2008) (ZHAO; WANG, 2009) (WEINI, 2012), e em pesquisas acadêmicas/científicas (LOPES, 2013).

Na renderização de imagens 3D, além do consumo elevado de recursos computacionais, é exigido um longo tempo de processamento para renderização das imagens. A renderização de imagens é um obstáculo na produção de animação e vídeo em grande escala (WEINI, 2012).

O desenvolvimento de tecnologias e ferramentas de software para processamento distribuído, como LoadLeveler (IBM, 2016), OpenPBS (ANL, 2004), contribuem na implementação de gerenciadores de processos distribuídos. As bibliotecas LAM/MPI (ANL, 2018) e Gearman (AKER, 2018) auxiliam na programação de softwares dedicados ao processamento distribuído. Exigem implantação e configuração complexa em ambientes dedicados para processamento distribuído, além de uma curva elevada de aprendizado.

Um projeto inicial deste trabalho, especifico para uso com Matlab, pode ser verificado em uma publicação de 2011: Prooceddings of XXXII CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering (BRITO, 2011). A novidade deste trabalho se destaca em disponibilizar uma ferramenta pré-configurada para o processamento de grandes massas de dados.

Dentro deste contexto, propõem-se implementar uma solução pré-configurada de *software* para processamento distribuído. Ao invés de criar um pacote de ferramentas, optou-se pela busca da simplificação da usabilidade para o usuário, dispensando a necessidade de programação em uma linguagem especifica, e reduzindo o tempo dedicado com a implantação.

De forma genérica, este gerenciador de processos distribuídos pré-configurado, permite executar qualquer *script* ou executável que suporte chamadas por linha de comando. Para este proposito o sistema agrega uma interface de monitoramento *Web* com um gerenciador de processos distribuídos. E para simplificar o processo de configuração e implantação, utilizou-se um Sistema Operacional com suporte de inicialização, *boot*, para memória RAM.

Neste trabalho a modelagem do projeto proposto ilustra o funcionamento do *software* desenvolvido baseado no paradigma mestre escravo (ILLINOIS, 1996) (HSU, 2007). Em seguida serão apresentadas as tecnologias utilizadas para implantação do gerenciador de processos distribuídos pré-configurado, e finalmente, apresentados os resultados dos testes realizados utilizando o *software* Blender (BLENDER FOUNDATION, 2018) para validar o trabalho proposto.

## 2. ANÁLISE DE REQUISITOS

- **Software:** Os requisitos definidos de software para implementação foram: Controle de processos pelo pid (Process ID, número identificador do processo, utilizado para gerenciar processos em um sistema operacional); funcionar em Sistema Operacional livre GNU/Linux; suporte do Sistema Operacional a inicialização para memória RAM; uso de chamadas remotas via RESTful (Implementação *webservice* baseada na arquitetura REST(*Representational State Transfer*) (RICHARDSON; RUBY, 2007), ou Transferência de Estado Representacional, arquitetura de sistemas abstraída da World Wide Web); balanceamento de carga de processos; uso de compartilhamento de arquivos em remoto NFS (Network File System - sistema de arquivos distribuídos em rede); interface de gerenciamento e controle via *Web* e linguagem de programação Python 2.7.
- **Hardware:** Requisitos mínimos de *hardware* suportados incluem: mínimo de 512MB de memória RAM, interface de rede de alta velocidade Gigabit, processador de 2 núcleos ou mais, placa mãe com interface USB e inicialização por interface USB.

## 3. ARQUITETURA DO GERENCIADOR DE PROCESSOS: MESTRE

A arquitetura do gerenciador de processos distribuídos pré-configurado proposto neste trabalho é composto por um nó mestre responsável pelas atividades de gerenciamento. Este tem a função de distribuir atividades para os processos escravos disponíveis na rede de computadores, conforme apresentado na Figura 1. A comunicação entre mestre e escravo é feita usando chamadas RESTful (RICHARDSON; RUBY, 2007) (FENG; SHEN; FAN, 2009).

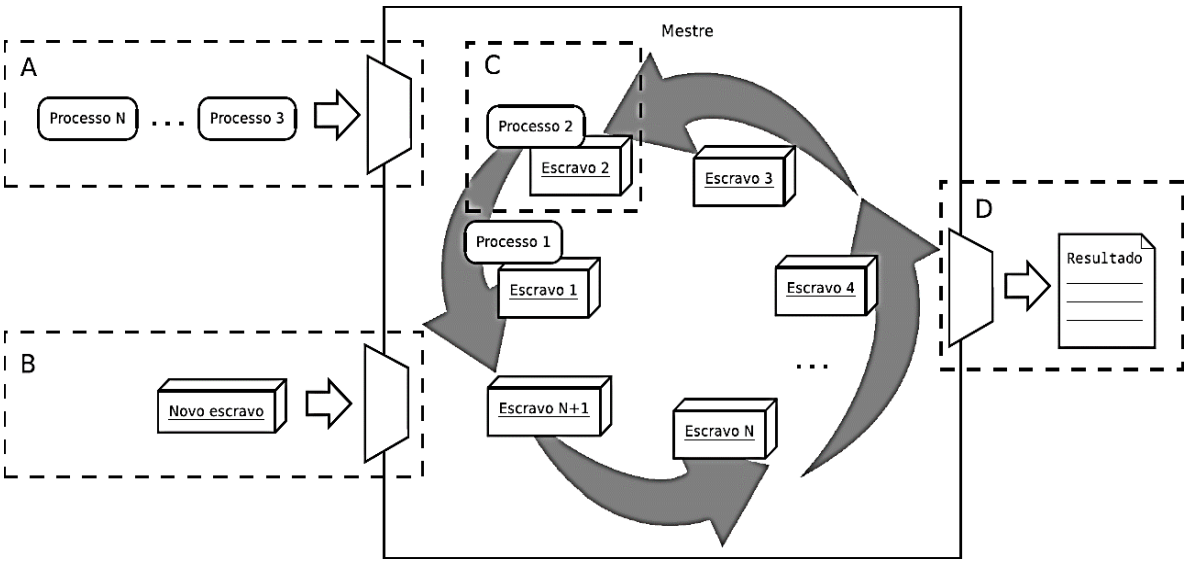


**Figura 1.** Escalonamento de processos.

Na Figura 1, pode-se verificar que a entrada de processos é realizada no quadro A, a entrada de escravos é localizada no quadro B, observa-se no quadro C a distribuição de processos para os escravos. O nó mestre é responsável pelo escalonamento, controle e acompanhamento de processos, visando que cada processo seja efetivamente executado. Os nós escravos são monitorados periodicamente, e em caso de falha na comunicação com escravos, as tarefas não executadas são redistribuídas para os nós escravos disponíveis. Por fim no quadro D tem-se a saída dos dados processados.

Os resultados gerados, quadro D, podem ser gravados em uma unidade de disco compartilhada pelos nós escravos, ou ainda registrados em um banco de dados conforme o processamento do *script* ou executável utilizado.

## 4. INTERFACE E COMUNICAÇÃO

O gerenciamento do sistema dispõe de uma interface *Web* para configuração e acompanhamento do processamento, podendo ser acessada remotamente por navegadores *Web*. É possível implementar novas interfaces para monitoramento dos nós mestre e escravo, devido ao uso da tecnologia RESTful. Prévia da interface pode ser visualizada no repositório Github deste projeto (DETTENBORN, 2018).

## 5. TOLERÂNCIA A FALHAS

A tolerância a falhas implementada, em relação a disponibilidade, segundo estudo de (TANENBAUM; VAN STEEN, 2007), busca garantir que novos nós escravos, possam ser facilmente adicionados ou excluídos durante o tempo de execução, sem que haja perda dos dados que estavam em processamento, pois os processos serão realocados pelo mestre para um nó escravo assim que ele estiver disponível.

## 6. FASES DE EXECUÇÃO

Durante o processo de execução, o nó mestre passa por três fases distintas, de maneira cíclica, conforme a Figura 2:

- **FASE 1** - consiste na verificação inicial do gerenciador de processos distribuídos, com a finalidade de detectar a existência de dados disponíveis para execução.
- **FASE 2** - é o momento onde o mestre distribui os processos a serem executados pelos escravos.
- **FASE 3** - é destinada a validação e detecção de erros, ou seja, verificar se os dados foram corretamente processados e se ocorreram erros.

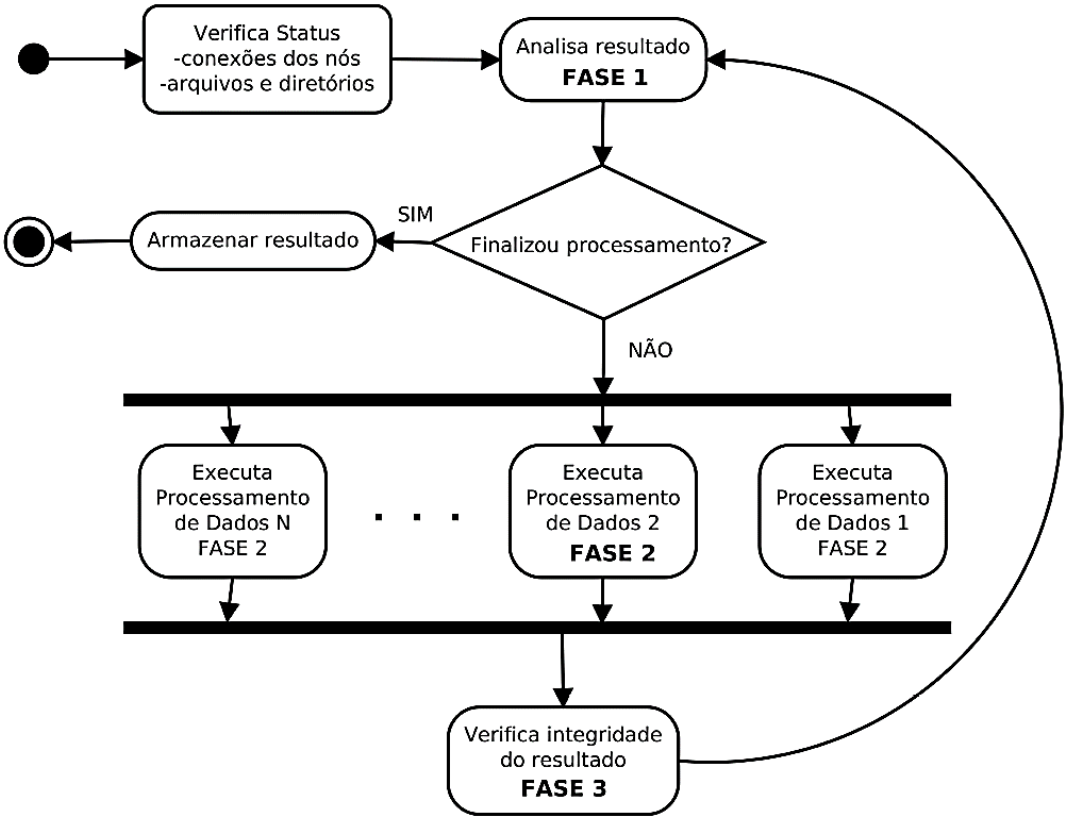


**Figura 2.** Fases de execução.

As fases 1 e 3 podem ser personalizadas pelo usuário, observando que devem conter a análise básica proposta no modelo que acompanha o software. A fase 2 é exclusiva para uso do administrador do gerenciador de processos distribuídos, que deve observar a sequência de parâmetros que o *shell script* (linguagem de programação utilizada principalmente em linha de comando para gerenciamento de sistemas operacionais) receberá do sistema durante o processamento, pois estes parâmetros que diferenciará a execução de cada processo distribuído. Mais detalhes sobre a configuração podem ser encontrados no repositório GitHub deste projeto (DETTENBORN, 2018).

**Parâmetros da Fase 2:** Na fase 2 é necessário informar qual será o *script*, executável ou binário. Também informar os parâmetros para diferenciar as chamadas que serão executadas remotamente em cada escravo.

Os parâmetros são informados no formato de estrutura de dados JSON (SEVERANCE, 2012). Trata-se de uma notação de dados feito na linguagem JavaScript, (FLANAGAN, 2011), que pode utilizar matrizes ou elementos em sua estrutura. O mestre recebe estas matrizes e executa o produto cartesiano entre elas para obter a lista completa de todos os processos.

## 7. NÓ ESCRAVO

A atividade principal do nó escravo é executar o script definido na fase 2 da Figura 2. Inicialmente, configura-se o nó escravo para permitir o estabelecimento de comunicação com o nó mestre. Os dados da configuração inicial são informados por uma interface Web, com instruções via RESTful, os dados informados são:

- O número máximo de processos que este escravo irá processar em paralelo. O valor padrão é dado pela contagem de núcleos que o sistema operacional detecta no hardware do nó escravo.
- O endereço IP e porta que o nó mestre está configurado, para estabelecer comunicação entre mestre e escravo.

## 8. SOFTWARE PRÉ-INSTALADO

Para auxiliar no processamento da criação do gerenciador de processos distribuídos pré-configurado, optou-se por pré-instalar o software em um sistema operacional Live USB, com aproximadamente 300MB, sem necessitar a instalação diretamente no disco

rígido. O sistema operacional adotado foi o Grml (GRML LIVE LINUX, 2018), que é baseado na distribuição GNU/Linux Debian (DEBIAN, 2018).

Neste contexto, com o uso de um dispositivo de armazenamento USB, é possível instanciar vários nós de processamento, diretamente para memória principal. Também é possível em uma rede local dedicada, com a tecnologia PXE (ORACLE, 2018) instanciar vários nós escravos utilizando o boot pela interface de rede.

## 9. VALIDAÇÃO DA FERRAMENTA PROPOSTA

Para avaliar o gerenciador de processos distribuídos pré-configurado optou-se pelo uso do Blender, que é um software de código aberto para modelagem, animação, renderização e edição de vídeo. Com o Blender é possível renderizar uma animação ou vídeo quadro a quadro com chamadas por linha de comando. Neste teste tem-se uma das cenas da animação Garoto Coisa (GOES, 2012), com duração de 40 segundos, 24 quadros de imagem por segundo, equivalendo a um total de 980 quadros.

Os experimentos foram realizados no Laboratório de Redes Avançadas e Multimídia - LABRAM da UFT, Campus de Palmas. Com 11 computadores com a seguinte configuração de hardware: DELL Optiplex 790 Quad Core i3-2120 CPU 3.30GHz 3292MHz, 3072 Cache e 8GB de RAM.

O cálculo do desempenho foi mensurado a partir do processamento em um único núcleo, e em seguida em mais de um núcleo no mesmo computador. A mensuração feita em um computador foi comparada com o processamento realizado no gerenciador de processos distribuídos pré-configurado proposto.

O tempo utilizado com o processo de implantação e configuração do gerenciador de processos distribuídos foi de aproximadamente 1 hora. Como foram utilizados 11 computadores, em média, utilizou-se 5 minutos e 30 segundos para configurar cada equipamento.

Ao todo tem-se 980 quadros de imagens que foram processados. Devido algumas características do Blender, é possível processar o intervalo e a quantidade desejada de quadros de imagens por processo.

No caso da granularidade, tem-se 980 processos (1 processo por quadro de imagem). Deve-se observar essa questão com a devida importância para que a distribuição de processos seja equilibrada entre os escravos. Por exemplo: para os 10

escravos, temos 980 quadros de imagens para serem processados, resulta-se em 98 quadros de imagens por escravo.

## 10. RESULTADOS DOS EXPERIMENTOS

Como se pode observar na Figura 3, quando é feito o processamento dos dados em apenas 1 computador, obtém-se uma pequena melhora no desempenho do processamento ao executar processos em paralelo.

Por outro lado, ao executar os processos no gerenciador de processos distribuídos pré-configurado, em mais de um computador, o tempo de processamento reduz de forma considerável: reduziu de 7 horas para aproximadamente 40 minutos.

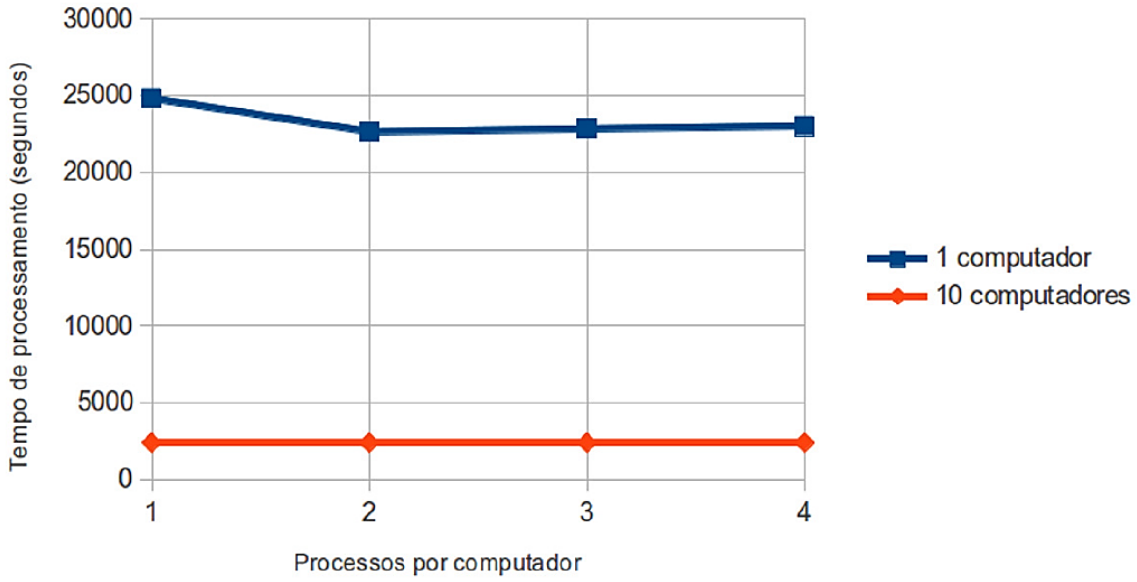


**Figura 3.** Comparativo entre execução local e gerenciador de processos distribuídos pré-configurado.

Sobre processamento paralelo, para o teste realizado no gerenciador de processos distribuídos pré-configurado na Figura 3 não se observou melhoria significativa no desempenho do processamento dos dados.

Observou-se um grande trafego de dados durante o processamento de dados no gerenciador de processos distribuídos pré-configurado, verificou-se que o aumento foi devido a comunicação entre mestre e escravos, e também em relação ao compartilhamento NFS.

## 11. CONCLUSÃO

Neste trabalho foi implementado uma solução de software para processamento distribuído pré-configurada via interface amigável ao usuário para gerenciamento de processos em um sistema distribuído.

Simplificou-se o processo de configuração e implantação do gerenciador de processos distribuídos pré-configurado, para suportar um ambiente heterogêneo com computadores de características diversas de hardware com o uso de um sistema operacional Grml, pré-instalado, com software de processamento distribuído.

De maneira geral, este gerenciador de processos distribuídos pré-configurado, permite executar qualquer script ou executável que suporte chamadas por linha de comando. Para este proposito tem-se uma interface de monitoramento e o gerenciador de processos distribuídos.

O tempo total utilizado para implantar o gerenciador de processos distribuídos pré-configurado foi pequeno devido ao uso do sistema operacional pré-instalado. O tempo total de processamento dos dados foi reduzido de forma considerável pela adição de nós escravos.

## REFERÊNCIAS

AKER, B. **Gearman [Gearman Job Server].** 2018. Disponível em: <http://gearman.org>. Acesso em: 02 jun. 2018.

ANL. Argonne National Laboratory: Math and Computer Science Division. **OpenPBS Public Home.** 2004. Disponível em: <http://www.mcs.anl.gov/research/projects/openpbs/>. Acesso em: 02 jun. 2018.

_____. **The Message Passing Interface (MPI) standard.** 2018. Disponível em: <http://www.mcs.anl.gov/research/projects/mpi/>. Acesso em: 02 jun. 2018.

BLENDER FOUNDATION. **Blender.org:** Home of the Blender project - Free and Open 3D Creation Software. 2018. Disponível em: <https://www.blender.org/>. Acesso em: 02 jun. 2018.

BRITO, George Lauro Ribeiro; DETTENBORN, Renê; BARBOSA, Gentil Veloso. **Ambiente Expresso para Processamento Distribuído com Matlab.** Prooceddings Of Xxxii Cilamce: Iberian Latin American Congress on Computational Methods in Engineering, Ouro Preto-MG, v. 32, 13 nov. 2011.

DEBIAN. **Debian**: **O Sistema Operacional Universal.** 2018. Disponível em: <https://www.debian.org>. Acesso em: 02 jun. 2018.

DETTENBORN, Renê. **GitHub - renedet/managerpp:** Gerenciador de Processos

Distribuídos. 2018. Disponível em: <https://github.com/renedet/managerpp>. Acesso em: 02 jun. 2018.

FENG, Xinyang; SHEN, Jianjing; FAN, Ying. **REST: An alternative to RPC for Web services architecture.** 2009 First International Conference On Future Information Networks, out. 2009. IEEE.

FLANAGAN, David. **JavaScript: The Definitive Guide.** 6. ed. [s.i.]: O'reilly Media, 2011. 1100 p.

GOES, Erick Henrique Silva. **Garoto Coisa A DIGESTAO.** 2012. Disponível em: <https://youtu.be/SMAj0YmDe8E>. Acesso em: 02 jun. 2018.

GRML LIVE LINUX. **Grml.org - Debian Live system:** CD for sysadmins and texttool-users. 2018. Disponível em: <https://grml.org>. Acesso em: 02 jun. 2018.

HSU, Ching-hsien; CHEN, Tai-lung; LI, Kuan-ching. **Performance effective pre-scheduling strategy for heterogeneous grid systems in the master slave paradigm.** Future Generation Computer Systems, S.i., v. 27, n. 4, p.569-579, maio 2007.

IBM. **LoadLeveler V5.1 documentation.** 2016. Disponível em: <https://www.ibm.com/support/knowledgecenter/SSFJTW_5.1.0/loadl.v5r1_welcome.html>. Acesso em: 02 jun. 2018.

ILLINOIS, University. **The Master-Slave Paradigm.** Parallel Programming Laboratory - Departament of Computer Science. 1996. Disponível em: <http://charm.cs.uiuc.edu/research/masterSlave>. Acesso em: 02 jun. 2018.

INTEL. **50 Years of Moore's Law.** 2018. Disponível em: <http://www.intel.com/content/www/us/en/silicon-innovations/moores-law-technology.html>. Acesso em: 02 jun. 2018.

LING, Huajun; GONG, Bin. **The Design and Implementation of Render Farm Manager Based on OpenPBS.** Computer-aided Industrial Design And Conceptual Design, 2008**:** 9th International Conference on, Kunming, China, p.1056-1058, 22 nov. 2008. CAID/CD 2008.

LOPES, Noêmia. **Unicamp tem segundo supercomputador mais rápido na América Latina, segundo ranking.** 2013. Disponível em: <http://agencia.fapesp.br/unicamp_tem_segundo_supercomputador_mais_rapido_na_america_latina_segundo_ranking_/17678/>. Acesso em: 02 jun. 2018.

ORACLE. **Understanding PXE Booting and Kickstart Technology.** 2018. Disponível em: <http://docs.oracle.com/cd/E24628_01/em.121/e27046/appdx_pxeboot.htm>. Acesso em: 02 jun. 2018.

RICHARDSON, Leonard; RUBY, Sam. **RESTful Web Services.** S.i: O'reilly Media, 2007. 454 p.

SEVERANCE, C.. **Discovering JavaScript Object Notation.** Computer**,** [s.l.], v. 45, n. 4, p.6-8, abr. 2012. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/mc.2012.132.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Sistemas Distribuídos:** princípios e

paradigmas. 2. ed. São Paulo-sp: Pearson Prentice Hall, 2007.

ZHAO, Yujiao; WANG, Zhengjun. **Research and Design of a Service Management System for Deadline Render Farm.** Environmental Science And Information Application Technology, 2009**:** International Conference on, Wuhan, China, v. 2, p.542-545, 4 jul. 2009. ESIAT 2009.

WEINI, Zhou et al. **A New Software Architecture for Ultra-large-scale Rendering Cloud.** 2012 11th International Symposium On Distributed Computing And Applications To Business, Engineering & Science, [s.l.], p.196-199, out. 2012. IEEE. http://dx.doi.org/10.1109/dcabes.2012.10.